Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

Blockchain Applications Beyond the Cryptocurrency Casino: The Punishment not Reward (PnR) Blockchain Architecture

Richard Banach*

[1]Department of Computer Science, University of Manchester, Oxford Road, Manchester, M13 9PL, UK

Correspondence
*Richard Banach, Department of Computer Science, University of Manchester. Email: richard.banach@manchester.ac.uk

Present Address
Department of Computer Science, University of Manchester, UK

Abstract

The Bitcoin model originated blockchain architectures and inspired their further development. Blockchain architectures are still most commonly associated with currency applications, and with financial speculation. Bitcoin's rewards for *Proof of Work* mining became the default consensus technique for blockchains.

As an alternative to *reward* mechanisms for blockchain maintenance, we propose *punishment* mechanisms for neglecting to maintain the blockchain (provided participants are intrinsically motivated to be beneficiaries of the blockchain application). Punishment not Reward (PnR) is convincing for enterprise mission critical blockchain applications, and potential punishment mechanisms include *denial of service* and/or *revocation of confidentiality*. This obviates the need for reward via cryptocurrencies, along with their attendant volatility, insecure ecosystem and market manipulation demerits.

The privacy concerns of competing entities participating in a blockchain application are *prima facie* in conflict with the needs of the community to be able to inflict punishment mechanisms. Conflicts of this kind can be addressed via sophisticated cryptographic techniques such as secret sharing, multiparty computation, zero-knowledge proofs, etc., which play a vital role.

We stress the importance of correctly balancing all *application specific* interests in the engineering of blockchain applications, so that the mix of incentives and disincentives is stabilising.

KEYWORDS:
blockchain, cryptocurrency, ICO, reward, punishment, consensus, denial of service, revocation of confidentiality, zero-knowledge proof, zk-SNARK.

1 | INTRODUCTION

The years 2016 and 2017 saw a veritable boom in interest in cryptocurrencies and blockchains. The public imagination was caught up in a drive to speculation in Bitcoin, and in cryptocurrencies in general. This caused a bubble in Bitcoin's value in December 2017. The result was predictable. Established players in cryptocurrencies had got out of Bitcoin and had consolidated their gains by December 2017, while recent incomers lost a lot of money.

The concept of blockchain, the technical foundation of Bitcoin, but with much wider potential for application[1,2], got caught up in this. Consequently, for a large swathe of the public, blockchain and distributed ledger technology (DLT) more generally, are conflated with cryptocurrency mechanisms, to the detriment of the understanding of the potential of the former (see, for example, the definition in[3]).

Cryptocurrencies are prone to a host of problems which we discuss below. The incisive title of the present paper is intended to highlight the opportunities for taking advantage of blockchain/DLT without the involvement of cryptocurrencies *provided the incentives are aligned correctly*. This possibility has tended to be mentioned rather *sotto voce* to date. The emphasis on incentive alignment is a point we underline strongly in this paper.

We contend that for a blockchain application to succeed in the longer term, all the potential incentives and disincentives of all the participants involved must be taken into account at the outset. Furthermore, among the various incentives, ones that stabilise the application in the longer term must be identified and must be engineered to be able to override any destabilising forces. We also propose that among potential disincentives, *denial of service* and *revocation of anonymity* can be employed as useful disincentives, counteracting the absence of a cryptocurrency as an incentivising mechanism, in a cryptocurrency-free application architecture. We use these insights to construct a generic *Punishment not Reward* (PnR) model for a class of blockchain applications that is widely applicable. The ability to inflict punishment by a blockchain community is intrinsically in conflict with individual participants' desire for confidentiality, and we describe how sophisticated cryptographic tools such as secret sharing, multiparty computation, zero-knowledge proofs, and others, can be brought to bear to address the contradiction.

Many of the individual details that we discuss in this paper are well known today in their own right. Our aim is not to claim that these observations are novel in themselves, but to deconstruct the swirl of ideas around blockchain and cryptocurrencies, with the goal of using these basic ingredients to assemble a model (our PnR concept) less prone to the problems that the usual arrangement throws up (and which have been commented on at length by now, particularly in the press and social science literature [4,5,6,7,8]).

The rest of the paper is as follows. In Section 2 we have a review of a number of aspects of Bitcoin, to elicit our own interpretation of its features for later reappraisal. Section 3 reviews Ethereum, though much more briefly, and Section 4 briefly reviews other developments, again for future reference. Section 5 discusses incentives and disincentives more generally, before in Section 6, we introduce the main proposal of this paper, the PnR blockchain application architecture, in which punishment can be used to motivate (good) behaviour in distributed systems in general and in blockchain applications in particular, just as reward via coins and tokens typically does these days. We chiefly have in mind enterprise blockchain applications, and for scaling and privacy reasons, we see these as typically being sector specific. In such cases, the startup phase is conceptually problematic, so we consider this in Section 7. In Section 8 we describe an outline system architecture to support the PnR application architecture, and in Section 9 we discuss a small case study, initially inspired by the legal sector. In both of these, the privacy/punishment contradiction mentioned above vividly comes to the fore, and is discussed in detail. Section 10 concludes.

2 | BITCOIN

As is well known, Bitcoin was launched in the shadow of the financial crash of 2008, bringing to life the proposal in [9]. Nowadays it is a textbook subject, e.g. [10,11]. As described in [10], Bitcoin was the culmination of a long sequence of attempts to invent a viable model for true electronic digital money that did not depend on trusting an individual third party. The breakthrough concept came about by combining a number of things.

- Firstly, there was the concept of an unbroken accounting chain of transactions involving a given unit of value (to prove absence of double spending) — the blockchain.
- Secondly, there was a distributed approach to the creation of the said accounting chain via cryptographically linked blocks of transactions (thus ensuring integrity in the longer term).
- Thirdly, there was the idea of having block creation delegated to 'miners' who were, in effect, randomly chosen per block (thus obviating the need for much trust in any specific individual), with the rest of the community checking the work to ensure it had been done correctly.
- Fourthly, there was the idea of motivating the miners to engage in block creation by rewarding them (using the self same currency that was at issue).
- Fifthly, there was the idea that the random assignment of the miner who would create the next block would be done by having the miners work competitively at hash breaking (the dependability of the hash function ensuring that the process was intrinsically fair — yielding the much debated *proof of work* (POW) concept [10,11]).
- Finally, the miners' reward would be *new* units of the currency, and mining would be the *only* way that the currency supply would grow.

It is vital to appreciate that these half a dozen or so ideas are actually independent. There is no logical necessity to conflate them, and what we do later in the paper depends on viewing them as separate ingredients, and not as an indivisible mix. For sure, they came together perfectly in the creation of Bitcoin, as its robust survival eloquently demonstrates, but for other purposes, their indivisible combination is not inevitable.

By the time Bitcoin grew into a phenomenon that reached the consciousness of the public at large (let us say 2016/17 for the sake of argument), POW had attracted attention for the appreciable quantities of energy that it consumed, as miners worked at hash breaking computations that were entirely unproductive (aside from when they yielded the right to create a block and to earn Bitcoins thereby). Commentaries differed in which country's electricity consumption to compare with the worldwide energy (currently over 70 Terrawatt-hours per year) expended on Bitcoin hash

breaking (Ireland's, Denmark's, Singapore's, Austria's etc.[1,6]). So POW (and by association, all the technologies involved with Bitcoin) became a *bad thing* in certain quarters of society; see e.g.[12].

The architecture of the Bitcoin blockchain is intrinsically decentralised. The absence of a single authority to dictate its operation and to dictate the value of Bitcoin inspired a lot of thinking that technologies like Bitcoin would usher in a new utopian era of worldwide cooperation, unfettered by powerful centralised interests that were capable of manipulating events to the disadvantage of the population at large. This was especially so in the shadow of the 2008 financial crash and its consequences, many of which were so detrimental to the average individual citizen in large parts of the world — such as the massive transmutation of privately incurred debt (by the banking system) into publicly discharged debt (sustained by taxpaying populations around the world),[13,14]. In a decentralised cryptocurrency there are no government imposed devaluations.

But the flipside of the idealistic decentralised utopia is the admission of anarchic and deceitful behaviours, unchecked by any authority powerful enough to impose discipline and the common good. As an illustration, we can mention that there are a hundred and eighty or so fiat currencies in the world, whereas there are around two thousand cryptocurrencies, a figure that can fluctuate quite widely, as cryptocurrencies come and go. How helpful to the average citizen is such a massive proliferation? We comment further on this point below. Another good illustration is the Bitcoin boom and crash of December 2017 itself. This was fueled by an extraordinary amount of market manipulation in the latter half of 2017, as evidenced by the huge surge in Bitcoin related junk email during that period (at least, as experienced by the author in his own mailbox). This promptly ceased, more or less, as soon as the crash had taken place.[1]

One of the less often noted characteristics of completely decentralised systems like Bitcoin, is that they are intrinsically unstable (unless there are steadying forces in place). Bitcoin is sufficiently decentralised that this instability has delivered noticeable effects. The facts that: (a) rewards (i.e. Bitcoins) are generated solely though hash breaking; (b) hashing algorithms are effective at generating outputs that are effectively random; (c) that because of (b), the more random trials a miner is able to generate per unit time, the greater the chance of a successful hash break, and thus of earning Bitcoins — implies that a small advantage (in terms of computing resources) leads statistically to greater rewards, and thus to the opportunity to invest in even more computing resources, leading to even greater rewards, etc.; i.e. we have a positive feedback loop. In the case of Bitcoin, the drive to greater hash breaking power has led to the creation of custom Bitcoin-hash-breaking ASICs[15], and to the fact that Bitmain[16,17], now controls a majority (or close to a majority) of the world's Bitcoin creation power. Decentralised system instability has led to Bitcoin becoming, to all intents and purposes, a currency that is at least close to being centralised! This is rather far from the egalitarian and libertarian impulses that drive so much enthusiasm for decentralised systems in general, and which are highly visible in many places, a prominent example being the social media platform Steemit[18].

The Bitcoin boom and crash of December 2017 was facilitated by the decentralised and unregulated nature of the Bitcoin network, which imposed no restraint at all on any activity within the network. This allowed the massive market manipulation seen in 2017, at least as measured by junkmail traffic. Despite the boom and crash, many people remain invested in Bitcoin, hoping perhaps for another boom. I personally doubt it (footnote 1 notwithstanding), history being the guide. Thus, there was only ever one South Sea Bubble[19], only one Tulipmania[20], only one Railway Boom[21], only one Internet Bubble[22], and so on. Each boom tends to consume the vast majority of the people who are receptive to being persuaded by the widely disseminated promotion of the current craze. When the crash comes, they are left licking their wounds when it proves to the case that they invested too late to benefit. Once bitten twice shy, it is only human nature that they should forswear further inducements to participate[23,5].

Still, such crashes are not always all bad (except for the unlucky investors). The Railway Boom left a valuable railway infrastructure behind. The Internet Bubble left behind massive overcapacity in the worldwide communications network infrastructure, which has proved to be a boon for the development of all the internet based businesses we see today (not to mention facilitating the plague of spam email). Bitcoin, whatever one's opinion about cryptocurrencies in general, and whatever its ultimate fate proves to be, has generated the concept of blockchain, whose potential in wider contexts has yet to be fully appreciated.

Of course, the hazards associated with the Bitcoin ecosystem, and with cryptocurrencies in general, are not confined to threats arising from external entities. It is sufficient to recall the story of Mt. Gox[24] in the case of Bitcoin, or the DAO in the case of Ethereum[25].

Despite what has been said, undoubtedly Bitcoin will sustain a value into the future. This is because there is one use case for it whose durability can be seen to trump most other considerations, namely the transmission of value at distance for criminal purposes, especially at modest scale. Criminals and others engaged in less than pristine activities will always need a means of value transfer that offers an appreciable degree of anonymity. Cash fulfills this purpose extremely well, being totally anonymous and self-securing (through the incorporation of many anti-forgery measures), but it has the disadvantage of being a physical medium, needing to be exchanged by a physical transfer process — the more so as these days the introduction of increasingly demanding KYC and AML regulations make the use of large quantities of cash for such purposes increasingly problematic when done at distance. A cryptocurrency achieves the same goal without the disadvantage of a physical transfer process, even if it

[1] Having said this, at time of writing, upswings in Bitcoin junk email are discernible from time to time, prompted most likely, by hopes of a repeat of late 2017. Most recently, quite a steep a rise in the Bitcoin price was attributed to Donald Trump's attitudes to international trade, followed by a substantial fallback. And so it goes on.

does so with a measurable degree of risk — and Bitcoin *de facto* remains the best established cryptocurrency[26]. Other studies of the criminal use case include[27,28,29].

Still, the degree of risk is real. The famous episode that resulted in the takedown of the Silk Road dark web marketplace[30,31,32] and the imprisonment of its founder Ross Ulbricht, underlines the fact that perfect anonymity on the web is an illusion. If the authorities are determined enough to invest the resources required they can deanonymise almost anything. It is simply another cost/benefit tradeoff to be evaluated. The techniques for breaking the anonymity of Bitcoin-like transactions are becoming increasingly effective and automated, which has led to a downturn in large scale criminality via Bitcoin[33]. Nevertheless, at more modest scale, this world is alive and well. In[34], vignettes were shown of medium scale drug trading facilitated by physical transportation by post, and financed using Bitcoin. Of singular note was how the world of such lower level trading had been able to remove itself completely from the environment of the street level drugs trade, and the extreme violence that characterises that world. So much so, that ordinary university students were financing their studies by discreet low level activity in this sphere — a 'wrap' could be ordered and delivered, door to door, significantly faster than a pizza. (It has to be pointed out though, that also discussed in[34], was the takedown of a significant cartel, who were operating in the same way at a larger scale, and who were awarded exemplary prison sentences as a result, most of whom originated from students and ex-students from the universities in the Manchester area.) At any rate, even if holding Bitcoin for investment or speculation were to lose its appeal, we see that Bitcoin would retain fitness for purpose in the criminal context.

The very fact that Bitcoin has managed to acquire a value quite spontaneously, without the intervention or support of any centralised party, led to the concept of the Initial Coin Offering (ICO). By inventing a new cryptocurrency which is offered for sale at a price its founders presume will be found appealing by investors, the founders hope that it will spontaneously appreciate in value through trading by an eager public, and that two desirable (from their point of view) consequences will ensue therefrom. Firstly that the (invariably large) block of cryptocurrency retained by the founders will acquire an appreciable value, making them 'wealthy'. Secondly that the fiat currency invested in the cryptocurrency by investors and the public will be made available to the founders for directly funding a novel business venture ... or for embezzlement. All these behaviours have been clearly visible in the ICO world.

As has been frequently noted, an ICO comes with no rights at all bestowed on the investor in exchange for the cryptocurrency purchased. In that sense, it constitutes a singularly risky kind of investment. The presumption is always that the cryptocurrency will appreciate in value, but the by now many cases in which this has failed to happen show that the promise attached to the ICO is often an empty one. The prevalence of ICO scams has added to the enormous proliferation of cryptocurrencies seen in the last couple of years[35].[2]

Two further caveats. Firstly, all that has been said above is predicated on the fact that cryptocurrencies are unregulated. But this is not, strictly speaking, entirely the case. The regulatory authorities of many jurisdictions take a keen interest in developments in the cryptocurrency world, even if, in many cases, little of a practical nature results from that at the moment (although see[36] and other speeches by Mark Carney). A widely expressed attitude is 'wait and see' (before committing to any specific regulatory structure). The landscape is further made immeasurably more complicated by the widely differing perspectives (among those jurisdictions that *do* take up a legal position) on what legal category cryptocurrencies should fall under. Some jurisdictions treat cryptocurrencies as currency, others as securities, others as commodities, others as real property, etc. See[37,38] for good discussions.[3]

Regulation is frequently a precursor to taxation, and the interest of various jurisdictions in cryptocurrencies in general is often fueled by a potential taxation back story, to be brought to the fore, should a significant proportion of their economies escape standard taxation trawlnets via the crypto route. If significant regulation were to impact the working of Bitcoin, undoubtedly its value would be affected, but since for the criminal community avoiding the consequences of regulation is a *sine qua non*, in the long term, that particular use case sustaining Bitcoin would ensure that a long term value for it would persist, even if it might not be the same value it might have in a fully unregulated world.

Secondly, we have the Bitcoin specific fact that total maximum quantity of Bitcoin is fixed in advance[9] and the value of mining an individual block is hardwired to keep decreasing as more of the fixed quantity is generated. In the end, the reward for block creation reduces to just the transaction fees for the transactions the block contains. This alters the incentive structure for mining very significantly. In this world, will mining remain economical when measured against the costs of the energy consumed, and how will this affect the value of Bitcoin? If mining becomes uneconomical, will Bitcoin become unviable as a value exchange medium because the mechanisms that sustain it dissolve? At this point it is too early to say, but it seems clear that the arrangement of interests sustaining the Bitcoin ecosystem will realign significantly as this epoch is approached, and the Bitcoin protocol may need to undergo significant re-engineering to keep it viable.

[2] Slowly, this situation is changing, with ICOs needing to be approved in various ways, either through new regulatory mechanisms, or by coin exchanges, which, by refusing to list the coin of an ICO that they do not trust, can stifle a new ICO at birth.

[3] My own opinion is that a new international legal category will have to be invented to avoid the lack of fitness for purpose that all the existing choices exhibit to a greater or lesser extent. Organisations like the G20 and ISO have such a line of thought on their agenda. See[36] again, and[39].

3 | ETHEREUM

While Nakamoto realised from the outset that the blockchain idea at the heart of Bitcoin would support the automated execution of suitably formalised contracts[9], typically referred to as smart contracts nowadays, the way they are captured within the Bitcoin protocol is rather awkward, and lacks the degree of flexibility users would expect from a usable system for supporting contracts.

Ethereum[40], pioneered within the blockchain framework a Turing complete formal language (Solidity) for formulating smart contracts, and for progressing their execution. The Solidity language[41,42,43], resembles a typical state based high level language, featuring a runtime stack, a heap, and inheritance mechanisms. The structure of a typical contract, consisting predominantly of state changing *function*s, resembles many a transition system oriented formal specification language, for example Event-B[44,45] (to mention but one such system). One consequence of this is the pioneering of approaches to *formal* verification of smart contracts and their functioning, e.g. works to be found in[46]. Of course such a development is not surprising since verification of the progress of smart contracts lodged on the blockchain is part of the wider verification responsibility of the blockchain protocol, regardless of whether or not formal techniques contribute to this.

The much greater accessibility of the Solidity approach has made Ethereum a very popular base for smart contracts.[4] Furthermore the standardisation of the ERC 20 protocol (and its enhancement in further standards like ERC223, ERC721 and many more[47]) has made Ethereum the default home for the torrent of new cryptotokens that have been issued during the last few years.

Ethereum uses POW (although alternatives are about to be released at the time of writing). That means that all the comments made about POW above apply almost verbatim to Ethereum too, at least for now.

4 | OTHER DEVELOPMENTS

The standard approach to blockchain and cryptocurrencies uses a single chain of transactions secured by common consensus using public key cryptography. It has long been recognised that a single chain and complex consensus protocols throttle throughput and lead to problems of scale. Additionally, the time lag between successive block creations (e.g. the 10 mins. of Bitcoin) leads to significant latency problems, especially when we contemplate fast moving commercial activities.[5]

The recognition of these issues has fueled an explosion of innovation to build blockchain architectures that ameliorate some of these issues. There is, by now, a large literature on alternative ways of organising the internal mechanisms of a blockchain to militate for some good behaviour or to mitigate against some perceived bad behaviour[48]. A large proportion of the discussions is concentrated on consensus mechanisms. Good accounts can be found in[49,50,51].

Throughput has also been addressed. By now there are a number of blockchain architectures that allow up to tens of thousands, hundreds of thousands or even a million or more claimed transactions per second. Notable systems include HPB[52], EOS[53], Elastos[54], DEXON[55], and many others. To catalogue all these developments would occupy a prohibitively large amount of space.

All of the above not withstanding, we have argued that there is just as much of a threat to dependable working from criteria outside of the blockchain mechanism as there might be from internal vulnerabilities. So a greater generic emphasis is needed on external aspects of the design of a blockchain system than is typically seen today.

5 | INCENTIVES AND DISINCENTIVES

In the sections above, we discussed many of the pros and cons of cryptocurrencies. This was chiefly in the Bitcoin context, from which we extrapolated. It became clear that taking into account the wider context, cryptocurrencies are subject to almost limitless vagaries of motivation of the participants in their ecosystem. This evidently undermines their suitability for *arbitrary* applications.

When policy in any sphere is driven using specific incentives to influence behaviour, the law of unintended consequences rules. Recent history is rife with instances of perverse outcomes spawned by the use of incentive structures to drive behaviour. The health service sphere provides many examples, e.g. in the US[56], or in the much revered British NHS[57]. To reinforce the point, here is a low level but genuine example from my own experience.

[4] Turing completeness is often touted as being responsible for this, but it seems to me that the higher level of expressivity of the language is a much more plausible explanation.

[5] On average, the VISA network processes 9000 more transactions per unit time than Bitcoin, at an energy cost of less than one half of 1% of that of Bitcoin.

A supermarket I use issues tokens as rewards for shopping there. They have their own loyalty card and credit card, which identifies me and records my shopping behaviour. Using the loyalty card mechanism, they also regularly offer enticements in the form of discounts and bonus points, but these are mainly targeted at merchandise categories that *I habitually do not buy*.

But if I do *not* use my loyalty card, remaining anonymous and paying by cash like a casual customer, at times I can receive money-off vouchers at the till to entice future visits and greater engagement. These are worth far more than what is usually offered to me via the loyalty scheme. How do you imagine that this has modified my behaviour?

A more well known example of unintended consequences is the famous Target pregnancy story [58], in which a father discovered his daughter was pregnant because the Target store was sending her coupons for baby products. This, a few months before she was due, was being prompted by her purchasing a clutch of products that Target's machine learning algorithms had deduced were strong indicators of pregnancy and impending birth.

Thus, one has to be careful in balancing the motivations of *all* participants, if an incentive driven approach to behaviour engineering is to succeed. Moreover, in any situation in which individual elements of a community derive benefit from actions that are the responsibility of all, one has to guard against the risk of the Tragedy of the Commons [59], in which all take the benefit but many eschew the responsibilities; and so the whole arrangement falls apart. Another angle on the same essential problem is the repeated invocation of the sucker's payoff in an iterated non-cooperative Prisoners' Dilemma type game [60,61]. The ICO world can be a fertile ground for these behaviours.

What is needed is an appropriate balance among the incentives and/or disincentives that motivate the behaviour of the individuals involved in any given activity. And for that activity to remain viable in the longer term, stability is also needed. We address ways of achieving this for blockchains next.

6 | PUNISHMENT, NOT REWARD

The above discussion of cryptocurrencies was intended to illustrate, how in the cryptocurrency arena, human motivation *directly* impacts the functioning of the system more than in almost any other digital sphere (the wider system security world, to which cryptocurrencies belong, being the main exception). The points we made argued that cryptocurrencies are prone to many sources of instability coming from the wider real world context in which they exist and function. This being so, it is evident that any blockchain application for whose working cryptocurrencies are central, will be affected by the same issues to a greater or lesser extent, and this in turn can undermine the viability of the application, even if the purpose of the application is not focused on currency *per se*.

The discussion points to the conclusion that cryptocurrencies, *to be used as the main driver for an application*, are poorly suited to the task, since the unreliability and instability that they are the source of, are almost unavoidable. The most obvious conclusion to be drawn then, is to do without them in blockchain applications. If we contemplate this however, we must still overcome the problems of motivating the maintenance of the blockchain and of balancing the interests of all the parties involved, which is what cryptocurrencies are traditionally used for.

We do not offer a universal panacea to the issue of blockchain incentivisation in this paper, but we address a particular class of applications for which a plausible argument supporting a cryptocurrency free scheme for sustaining a blockchain solution can be constructed. The essential elements of the scheme are as follows.[6]

- The entities involved in the scheme must be such that any loss from reputational damage far outweighs any gain to be had from gaming the system. The potential consequences of loss of reputation following from misbehaviour —typically needing to be severe enough that they lead to organisational collapse— are the major stabilising force motivating good behaviour. Typical examples of entities of this kind would include: health service organisations, educational organisations, professional services such as the law and accountancy, commercial organisations for whom ethical considerations form an integral component of their public image, etc.

- The scheme is focused on running an application and enterprise specific permissioned blockchain. In this manner, all entities involved have a similar cost/benefit spectrum of incentives and disincentives for participation, and this makes it easier to evaluate whether the balance between these is appropriate. A specific application focus for all participants also resolves contention about what the criteria might be for defining behaviour that is bad enough to merit punishment (see below). Moreover, the major scalability challenges of running a massive all-purpose blockchain are avoided.

- The blockchain based scheme must support a mission critical part of the participant organisations' working, or at least, must support a functionality whose forfeit would result in a serious loss of some kind.

[6] A brief presentation of the following ideas appears in [62].

- Ideally, the immutability properties of the blockchain give a significant non-repudiation added value to automated processes in the organisations' working that would be hard to achieve by other means. Concomitantly, all entities involved must agree to accept the consequences of the inability to forget old information on the blockchain (especially given the implications of regulatory developments such as GDPR in the EU [63], or CCPA in California [64], etc.). Similarly, any application specific requirements for auditing [8], must be explicitly built into the blockchain protocol from the outset.

- If particular parts of the application specific blockchain protocol are best served by micropayments to participating entities, these can be managed by cryptocurrency-like micropayment mechanisms on the blockchain, on the understanding that the payments refer to fiat currency payments between specified entities, and are intended to be settled in bulk periodically, when the accumulated amounts justify fiat currency transaction charges and processes.

- The protocol managing the blockchain application must monitor participants' behaviour in respect of them discharging their obligations to maintain the viability of the blockchain. Failure to discharge agreed obligations can result in sanctions such as **denial of service to the application** and/or **revocation of anonymity** (among other, potentially application-specific measures), thus impacting on defaulting participants' viability. The tension between knowing participants' identities (so they can be punished if need be) and keeping details of participants' business appropriately confidential must be explicitly considered and resolved.

- While the protocol managing the blockchain application must cater for a range of expected eventualities in the progress of a smart contract, it is unlikely that every possible contingency can be programmed at the outset, so there should be recognised error exits in the protocol machine to allow for unforseen circumstances to be handled by exiting automated working, and having recourse to human mediated legal or commercial techniques.

We see in the above a balancing of incentives and disincentives that is not founded on cryptocurrencies (except for the trivial provisions of the third to last point). Instead of hoping to promote good behaviour by disbursing cryptocurrency, with the accompanying risk of the system being gamed by ingenious (or simply fraudulent) exploitation of the reward protocol,[7] we punish bad behaviour by withholding access to the application itself, or by revocation of degrees of anonymity, which participating entities would presumably find undesirable (or else they would not be participating in the first place). The other advantage of this approach is that it is founded on mechanisms that are purely *internal* to the application (relying on participants' presumed desire to benefit from using the application). This is in contrast to trying to foresee threats which in large measure are based on *external* circumstances that arise because of the presence on the chain of something (the cryptocurrency) that is of potential value to everybody. The internal focus implies that with *one* internal punishment mechanism we are able to counter potentially *many* threats to an externally appealing reward mechanism (by eliminating the need for it).

The need to be able to enforce denial of service impacts the kind of protocol that is used for driving the blockchain. Some kind of committee based policing is needed to approve normal participation by conforming entities and to deny access to rogue entities. A natural candidate for this is a suitable variation on Delegated Proof of Stake [66], as used in EOS [53], and many other systems by now. Other consensus protocol candidates that could provide a basis for the kind of blockchain we have in mind could be Proof of Authority [67] or Algorand [68]. Hyperledger [69] (and Hyperledger fabric in particular among the various Hyperledger projects [70]), is a framework within such approaches could be housed — as are Catayst [71], Jelurida [72] and others, newer entrants to the blockchain protocol family.

The above is a high level architectural account of the PnR idea. It is light on detail for two reasons. Firstly, we wish to stress the primary importance of balancing (dis)incentives as an overriding concern in designing a blockchain system — the human element will win out in the long run if it is not properly taken into account at the outset. If the (dis)incentives are inappropriately balanced the application will most likely not succeed, no matter how well the internal details are engineered. Secondly, one size will not fit all. Applications to which the PnR idea can be adapted will vary widely in detail, and the different parts of the architecture will need to be adapted to meet anticipated variations, in a properly designed system.

7 | THE STARTUP PHASE

When a typical blockchain based application or service is first made live, one of the most common problems encountered is a dearth of 'hashing power' (or its corresponding analogue, according to the blockchain architecture being used). The lack of diversity in the community of miners (or analogues) supporting the application or service undermines the perceived robustness of the decentralised system and the degree of trust that can be placed in it. This phenomenon is already visible in the real world, see e.g., [73].

[7] One might argue that because any protocol consists of a finite number of steps, it can only make provision for a finite number of kinds of eventuality, and that therefore, exploits might always exist that are able to look ahead more deeply than the protocol is able to do. The argument is a kind of mirror image of Lamport's classic nonexistence proof of the absence of a solution to the Byzantine Generals' Problem in the absence of any guarantees [65].

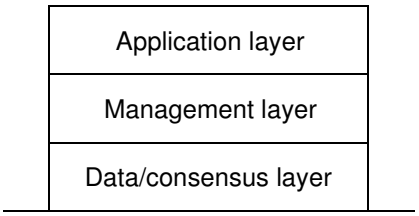

FIGURE 1 Outline system architecture to support the PnR blockchain scheme.

In the PnR scheme just described, the problem is, if anything, exacerbated by the intention to keep the blockchain permissioned and (relatively) special purpose. It is quite likely that the launch of a new PnR blockchain application will involve only a single entity. How then do we obtain the trust that is characteristic of large blockchains in such a situation? We offer a couple of suggestions.

Firstly, the single entity launching the application will be motivated to see it working properly. In a context such as we have stipulated, in which reputational loss is severely damaging for an organisation, launching an application for the purpose of subverting it makes little sense. So, even though with one sole participant there can be no 'committee' verifying good behaviour, the possibility of bad behaviour can be ignored with a relatively low level of risk.

Secondly, as soon as a second, competitor organisation joins the application (motivated by the same goals that led to the application's creation in the first place), the presence of two rival participants in the blockchain dramatically increases the intrinsic level of trust, since each, in effect, monitors the other. Additional organisations joining the fray rapidly increase this intrinsic level of trust, and the whole process can be accelerated if a consortium of enthusiastic early adopters can be recruited from the get go.

Thirdly, the participants joining the application, up to some threshold number, can be required to maintain a small stake in a widely established blockchain, e.g. Ethereum. Periodically, they carry out an Ethereum transaction in which they pay their stake back to themselves (modulo the transaction fee), and included in that transaction data, is a hash of the PnR blockchain's latest block. In this manner, the irrefutability of the fledgling application's blockchain can be supported by exploiting, somewhat parasitically, a *de facto* established blockchain in the early days of the PnR blockchain. Once the latter has accumulated an application specific threshold number of participants, this mechanism can be relinquished.

The exploitation of existing established blockchains is already used as a mechanism for increasing throughput and responsiveness for a number of cryptocurrencies, these being pegged in various ways to Bitcoin or Ethereum, etc. High transaction rates are maintained off the main chain, with reconciliation on the main chain taking place periodically. RSK[74] and the Lightning Network[75] on the Bitcoin chain and the sharding anticipated with the Constantinople and St. Petersburg Ethereum hard forks[76] are two examples among many of systems that use such mechanisms.

In the wider sense, the use of such established blockchains to provide more general purpose facilities (such as randomness, for example) in return for transaction fees, would provide a useful service for the kind of smaller, application specific blockchains we are discussing. We can see this as the first step in using established blockchains as the source of *Trusted Third Party* services of a general kind, which, although needing trust by the PnR blockchain (in the present instance) to supply the service, do not rely on a single preassigned entity to supply the service. This precludes the threat that such an entity, were it to know in advance that it is going to be asked to supply the service, might, for it own ends, be tempted to try to invest resources to subvert its PnR blockchain client in some way.

8 | AN OUTLINE SYSTEM ARCHITECTURE FOR PNR APPLICATIONS

In this section we present an outline system architecture to support the PnR blockchain scheme described above. Given how much we stress the extent to which such systems need to be application dependent, this outline is relatively low on detail, and there will be many options, both published already and as yet not envisaged or described, for fulfilling the requirements of each layer of the architectural model. However the model provides a top level design structure that can usefully separate concerns, and we illustrate its use in a specific example in the remainder of the paper. Fig. 1 illustrates the architectural model.

8.1 | The Three Layer PnR Model

Our system architecture outline has three layers. At the lowest level is the data/consensus layer. Data is envisaged to be held in four types of storage: (a) in archival storage — this is mainly for forensic purposes in case of dispute resolution; (b) in traditional databases, either centralised or

replicated in standard ways — hashes of such data may or may not be held on-chain, according to requirements; (c) in permanent replicated indelible storage such as the interplanetary file system (IPFS)[77] — hashes of such data are held on-chain; (d) in transaction records posted on-chain.

The consensus part of this layer defines the mechanisms for achieving agreement about the serialisation of transaction records posted to the chain and for the actual updating of the chain. As discussed earlier, some variant of committee monitored rotating chairman style of consensus determination is foreseen as the most promising for the type of application we have in mind, but in principle, any consensus algorithm could be used.

The middle layer is the management layer. This defines the mechanisms for initiating the blockchain and keeping it running. This includes tasks such as participant enrollment, management of pseudo-identity mechanisms, management of participant good standing data, and the management of the protocol's punishment mechanisms. Also falling under the management layer's auspices are membership of the current management committee, assignment and reassignment of the appointed chair, archival of old on-chain data, etc.

At the top is the application layer. It contains the facilities that enable the 'business logic' of the system. To the extent that these involve transactions posted to the chain, this layer is prescriptive, in that it defines *what information* can be posted to the chain and under what circumstances. It is (necessarily) *not* prescriptive about off-chain activities of participants, although, given that the chain is intended to be relatively application specific, it will contain recommendations about what such relevant off-chain activities might be expected to look like.

8.2 | Management Layer Challenges: Knowing and not Knowing, Simultaneously

The biggest challenge in creating a specific architectural model such as this is lies in the management layer. Our proposition is that there is a community of participants on the chain *Comm*, who, in principle, are in competition with one another. In a blockchain context, this implies the maintenance of privacy/secrecy. And our further premise is that bad behaviour by a participant *P* earns some punishment, inflicted by the remaining community *Comm – P*. In a digital context, punishment would imply the withdrawal of some privilege Pr_P that is enjoyed via some digital means which we refer to as M_P. Those means must be confidential to *P*, otherwise rivals could undermine the relevant privilege Pr_P arbitrarily. On the other hand, the possibility of withdrawal of the privilege implies knowledge of M_P by the same rivals, otherwise they could not withdraw the privilege when the situation demanded (since we must assume that the miscreant *P* would not divulge M_P willingly to *Comm – P*). **This is a contradiction, since members of** *Comm – P* **cannot both know and not know** M_P**.**

From a fundamental point of view, the contradiction just stated cannot be avoided absolutely. The argument is simple. The crux of the matter is the secret M_P. In order that *Comm – P* acquire the right to punish *P*, should it become necessary, they need some access to M_P. On the other hand, excessively easy access to M_P can lead to the temptation to undermine the privilege Pr_P, and thereby to undermine *P*'s effectiveness. One approach to addressing this issue is for *P* to give access to M_P in an obscure manner **Obs**(M_P), from which it is not easy (although not impossible) to infer M_P itself.

The non-impossibility of gaining access to M_P from **Obs**(M_P) must be demonstrable. Otherwise, if *in extremis* the **Obs** mechanism is sufficiently opaque that it is in practice impossible to infer that M_P can be extracted from **Obs**(M_P), then *P* could offer nonsense data instead of M_P, obscured using **Obs** (or even nonsense data created without employing **Obs**), and then *Comm – P* could not mete out the required punishment, should it be needed. But if the **Obs** mechanism is **in**sufficiently opaque, then there is nothing to stop dishonest elements in *Comm – P* from investing the resource needed to extract M_P from **Obs**(M_P) and thereby to undermine *P* as described.[8] So we recover the earlier contradiction. However, our suggested approach has interspersed significant computational effort between the knowing and not knowing branches of the contradiction, which, we suggest, improves the situation. Clearly, the approach just sketched opens the door to a wealth of innovative designs for suitably effective **Obs** mechanisms. And one key ingredient that we can expect to commonly find in such mechanisms is a means of demonstrably asserting evidence or knowledge of some fact without revealing the fact itself.

These days, the gold standard for the convincing divulgence of knowledge of something, without revealing the something in question, is the zero-knowledge proof[78,79,80]. Zero knowledge proofs originated in the late 80's[81,82] and typically turned NP queries into related queries that a *verifier* could demand that a *prover* should demonstrate. These related queries were such that their correct discharge (as judged by the verifier) could not be accomplished without the prover's knowledge of the answer to the original query (barring a negligible chance of guessing correctly). Zero-knowledge proofs were something of a hammer looking for a nail, until blockchain, with its need to 'publicly check' that which is 'privately transacted', came along. Zero-knowledge proofs are currently attracting increasing interest in the blockchain world, with companies like qed-it[83] and appliedblockchain[84] adding these technologies to their blockchain products.

[8] The same argument can be used to deduce that extracting M_P from **Obs**(M_P) must require the cooperative effort of **several parties** —to prevent a single rogue entity from being tempted to maliciously undermine the activities of a competitor— on the presumption that building a conspiracy to do so is considerably harder than having one entity choosing to do so unilaterally.

For convenient non-interactive zero-knowledge checking of asserted truths (which otherwise requires the verifier to make online random choices of the subservient queries that the prover must discharge), the **zero-knowledge Succinct Non-interactive Argument of Knowledge (zk-SNARK)**[85,86,87,88,89] is, by now, the standard mechanism. This replaces the online random choices made by the verifier, with pseudo-random choices made by an unbiased pseudorandom number generator. In this manner, we can expect zk-SNARKs to figure prominently in **Obs** designs. These days they can be seen in action in the Monero and Zcash cryptocurrency systems[90,91]. Moreover, the Ethereum blockchain is also exploring deployment of zero-knowledge techniques, based on the Zcash and other systems[92].

To further dissuade individual rogue elements in *Comm – P* from undermining *P* by accessing M_P inappropriately, we can make the **Obs** mechanism depend on other, clever, cryptographic mechanisms invented over the years. We mention a few potential candidates now, without any claim about their potential utility for privacy preservation in blockchain applications in the future.

Secret sharing[93,94,95] allows a secret to be divided into portions that are distributed among a number of entities in a way that prevents a subset of the entities from knowing the whole secret. Group signatures[96,97,98] and ring signatures[99,100,101] are schemes that allow, in various ways, data to be signed by a single entity which is part of a group, but without revealing which member of the group actually signed the data. Multisignature schemes[102,103,104] are a generalisation of both of these that allow (for instance) n out of m signers (where $n \leq m$) to authorise some action. Multiparty computation[105,106] does similar service for computations more generally — no entity can complete the computation without the cooperation of the others. Homomorphic cryptography (e.g.[107] and references therein) allows computation to be done directly on encrypted data, without having to decrypt it first (thereby never revealing the data being worked on, nor the keys needed to decrypt it). And we should not dismiss, either, approaches that employ the TTP-via-blockchain possibilities mentioned above. We do not pretend to explore all such possibilities in this paper, but merely hint at directions for possible future innovation in this space.

Techniques such as those mentioned hold out the promise of forcing the undermining of **Obs**(M_P) to depend on conspiracy rather than individual misbehaviour, thus increasing the strength of the **Obs** mechanism that protects M_P. Further inventiveness in the construction of usable **Obs** mechanisms can be anticipated in future.

The case study in the latter part of this paper features a fairly simple **Obs** mechanism, for purposes of illustration. The privilege M_P in this case amounts to the real identity belonging to the pseudo-identity *P*, and the privilege owner posts a partial divulgence **Obs**(M_P) of M_P, capable of being completed to a full divulgence, but only with sufficient difficulty. The claim that the partial divulgence is honest is supported by a suitable zk-SNARK. In case of a sufficient transgression by *P*, the community can invest the resources to complete the construction of the full divulgence of M_P from **Obs**(M_P), and thereby to remove the relevant privilege from the real participant masquerading as *P*, thus inflicting the punishment.

The preceding illustrates the nontrivial challenges that the management layer needs to meet, which go considerably beyond the usual housekeeping jobs that such a layer has to handle (typically membership of the management committee, election of its chair, etc.). The management layer has to be receptive to what constitutes both good and bad behaviour by participants, and has to be able to react to the latter. The latter covers both relatively obvious non-compliance with protocol expectations, but also potentially includes an application specific remit within wider society, which makes the challenge the greater. A simple example is given in the case study below.

8.3 | Bitcoin (and Cryptocurrencies Generally) from a PnR Perspective

We can see in the above also a reflection of the incentives around the Bitcoin protocol, discussed in Section 2 and further elaborated subsequently, although Bitcoin was hardly conceived with PnR in mind. In the Bitcoin context, the privilege is the Bitcoin possessed by an owner. The means to the privilege is the private key of the owner. The partial divulgence is the (inadvertent) logging in the internet infrastructure of the internet activity of the owner when engaging with the Bitcoin system (although this is obviously not a deliberate deposition by the owner). The transgression is whatever the law enforcement authorities consider it so to be, i.e. it is unlawful activity facilitated by the owner's cryptocurrency holdings (albeit that this is evidently also outside the Bitcoin system itself). And the full divulgence, at considerable cost, is the internet forensics that, with effort, can reveal the real identity of the Bitcoin's owner.[9] So we can say that the Bitcoin protocol, viewed appropriately, exhibits the characteristics of a blockchain system that can impart *punishment as well as* reward. Evidently, the same remarks can be applied to any cryptocurrency.

9 | CASE STUDY: PAYMENT LOCKUP

A very standard practice in the business-to-business world is for goods and services to be provided, but for the due payments to the suppliers of the goods and services to only be made in arrears. Many smaller enterprises struggle to stay solvent because they do not receive due payment promptly enough, and correspondingly, many large enterprises exploit their power over smaller suppliers by withholding payment for an undue

[9] The episode that resulted in the takedown of the Silk Road, mentioned earlier, attests well to this.

length of time, improving the appearance of their own balance sheet by doing do. The delay between payment becoming due and payment being made is referred to as payment lockup, and it constitutes the focus of the case study here.[10] In fact, this kind of situation is the source of many ingenious proposals in the Fintech sector[108,109]. Below, we propose a simple blockchain architecture to defriction a scenario like this. However, before we do so, we make the following observation.

While it would be impossible to discuss the matter in full generality, broadly speaking, litigation is a public process, and so, in a legal sense, contracts are public documents. If their execution descends into dispute, the resolution takes place by means which are more or less public. On the other hand, in the vast majority of cases, contracts do not result in dispute, and the parties involved vastly prefer that the details of the contracts they undertake with each other remain private to themselves. One very common way of achieving this is for the parties to agree at the outset, to be bound by the decisions of arbitration panels, sitting in confidence, in case that a dispute arises between them. In the world of traditional paper contracts these circumstances are easy to arrange. However, in the world of smart contracts residing on a blockchain, it is the duty of the blockchain architecture to apply verification procedures to transactions that are intended to be placed on the chain. This means that the contract details that they belong to must be public to the extent that arbitrary blockchain verifiers must be able to access the contract and to determine the conformance or otherwise of any transaction posted against it. This in turn necessitates a deep reappraisal of what such contracts could contain. Rather than simply including in a smart contract all the information relevant to the proposed collaboration between the parties enacting the contract, there has to be a detailed examination and reassessment of all this information, and a detailed rationale for what part of the information may reside on the chain, what part is to remain private, and how these parts are to relate to one another. All of this will be application dependent as well as blockchain architecture dependent.

9.1 | Payment Lockup Application Outline

In this section we outline a simplified blockchain protocol *PayLck*, intended to address the kind of payment scenario alluded to above. We describe here the steady state condition, where sufficient participants exist to make the anonymisation provisions of the protocol 'reasonably effective' (*c.f.* the comments made earlier about the anonymity, or lack of it, in systems running over the internet). We address the boundary cases, such as startup, in the more detailed treatment below.

The participants are *S*uppliers and *C*ustomers. The blockchain is a permissioned chain managed by the suppliers (although variations in which high volume customers also choose to assist in the management burden, incentivised perhaps, by more lenient settlement terms, can certainly be contemplated). The management of the blockchain protocol is done via some variant of delegated proof of stake, or of one of the other committee based schemes discussed earlier, with a chair that is one of the suppliers, monitored by a management committee. Both chair and committee members are changed regularly (and preferably randomly), selected from among those members of the supplier pool that are in good standing with the chain.

A supplier *S* registers with the chain using its real identity and credentials. This information is held off-chain in a place accessible to the current chair and management committee. The supplier chooses a (temporary, random) private pseudo-identity and computes the corresponding public pseudo-identity. The latter is sent as part of *S*'s registration data. A valid-pseudo-identity transaction containing the public pseudo-identity is posted on-chain by the chair.

A valid pseudo-identity of its owner in good standing with the system bestows two basic rights. The first is the right to generate further pseudo-identities belonging to the same real identity and credentials. The proliferation of pseudo-identities that this generates aids in obscuring suppliers' activities from their competitors on the chain.[11] The second is the right to post and execute smart contracts on the chain. For a given application specific blockchain of the kind we are advocating, there will be a limited number of contract schemas, and each contract instance will be one of these, instantiated in the appropriate way. The amount of flexibility permitted in configuring a contract schema into a concrete contract instance will be application specific in general. For purposes of illustration, our payment lockup example protocol *PayLck* will be very simple, and the state machine for it appears in Fig. 2. It works as follows.

Supplier *S* and a customer *C* agree that *S* will supply goods/services to *C* under an instance of *PayLck*. None of the detailed terms of this, nor of the payment to be made upon completion are the business of the blockchain at large, and so are encrypted using a key known to *S* and *C* and are parceled up in digital paperwork stored in a durable off-chain medium, such as the IPFS[77]. A transaction confirming the agreement is posted on the chain. Each subsequent step of the execution of the smart contract instance proceeds in an analogous manner, i.e., whatever off-chain work

[10] The account in this paper is inspired by the situation in a law practice, the overwhelming bulk of whose work comes from business-to-business legal services.

[11] The same proliferation opens the door to an insider DOS attack of course, but the restricted scope of the permissioned chain is intended to ensure that there is little incentive for participants to launch such an attack, and variations of the design may be created to mitigate this possibility. We will not consider it further in the present simple model.

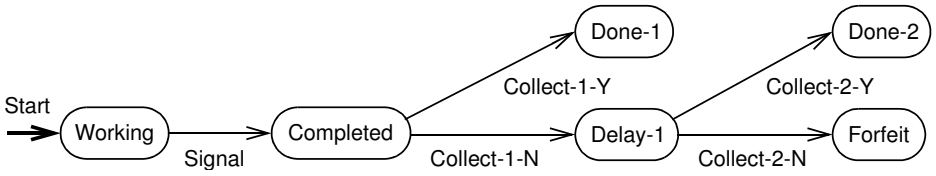

FIGURE 2 Outline *PayLck* smart contract state machine. *ExceptionExit* transitions to human mediated resolution procedures from all states except Done-1 and Done-2 are not shown.

is required at the relevant step of the protocol is done, and a record of it, typically containing the hashes etc. of the off-chain work is posted in an on-chain transaction, recording progress.

The *PayLck* application state machine is shown in Fig. 2. A *PayLck* instance starts by entering the Working state, with all details recorded in IPFS. *S* commences doing the work stipulated. When it is ready, *S* posts a Signal transaction which contacts *C* off-chain to announce that the work has been completed, and that payment will be claimed within the period agreed at the commencement of the contract, allowing *C* to ensure the funds are available for collection by *S* in the bank account also agreed at the commencement of the contract. If the funds are there after the allotted time, *S* collects the payment and posts a Collect-1-Y transaction, completing the *PayLck* instance in the Done-1 state. If the funds are not there, *S* posts a Collect-1-N transaction, and enters the Delay-1 state, allowing *C* an extension of the settlement period in return for a reduced discount on payment. The transactions Collect-2-Y and Collect-2-N complete the *PayLck* instance, except that after Collect-2-N, the Forfeit state indicates loss of any early settlement discount and the reversion to normal human-mediated settlement methods.

In the startup phase of *PayLck*, there may be only one supplier *S*. In this condition, *S* plays all the roles in the blockchain system. We assume that *S* has many *C*ustomers, so the provisions of the *PayLck* protocol are not completely redundant, since they can conceal customers' identities (as well as the details of the payments made). Besides this, the artifice of pegging to an established blockchain can be used to assure immutability of the chain contents while the number of participants remains small.

The blockchain protocol supporting the *PayLck* application insists that all transactions posted have a validity deadline. This enables a sufficiently old suffix of the chain to be decanted to archival storage, making the live chain easier to manage. The latter includes service transactions such as posting the register of valid participants that are in good standing (or not) with the chain. Since transactions such as this also have deadlines, they need to be refreshed regularly. Attending to this is but one part of the good standing duties that participants need to fulfill when elected to management committee roles, especially the chair role. The good standing provisions can be made subtle enough that excusable lapses, such as those due to expected or unexpected system recoveries/upgrades can be allowed for (although we do not explore nontrivial incarnations of such ideas in this paper). Of course, all interactions with the chain by off-chain entities, especially by the oracles that progress smart contract instances automatically, need to be carried out in a secure manner, supported by services such as Oraclize[110] (as an example of a service offering relevant facilities at the present time).

9.2 | The Data/Consensus Layer, Utilities

As stated, we assume that this example runs under a consensus algorithm such as delegated proof of stake. We do not include the details of this, but simply assume that under committee scrutiny, the chair assures that from the pool of candidate transactions posted to the chain, a strictly serial sequence of valid transactions is recorded in the blocks of the chain, which can be read by any participant in the system. We also assume that the consensus mechanism is *fair*, in that a candidate transaction that remains valid indefinitely, is eventually recorded on the chain.

We assume there are four kinds of data repository:

- ARCH — an archival storage medium, intended primarily for forensic use;
- CDS — a conventional data store, supporting modification and deletion as well as insertion;
- IPFS — a replicated distributed indelible data store for use with on-chain material;
- CHN — the blockchain itself.

The data layer provides methods for posting data to these repositories: **PostARCH**(...), **PostCDS**(...), **PostIPFS**(...), **PostCHN**(...). There are also methods for retrieving data from these, e.g. **FindARCH**(...) etc, and for modifying and deleting data in CDS. Needless to say, access to ARCH and CDS must be secured by conventional means and access logs must be maintained. Also, CDS should be held in encrypted form, with the key held

by the committee, with a notice of every access posted on the chain, so that inappropriate use of CDS can be detected. When the committee is reassigned, CDS is re-encrypted with a fresh key which is issued to the new committee.[12] Access to IPFS and CHN is protected by the mechanisms inherent to the deployment of cryptography in these systems.

We list some utility functionality needed in the rest of the system.

- *MAXLIFE* — the maximum validity lifetime of any transaction posted on-chain (after the expiration which, any of its remaining provisions must have beeen refreshed in a new transaction, allowing all transactions older than *MAXLIFE* to be archived).
- *H* — a known preimage resistant hash function producing 256-bit outputs (e.g. *H* is SHA-2 or SHA-3, suitably deployed).
- *RAND* — a source of public pseudo-random bits, generated, for example, as follows: *H* is applied to the most recent Ethereum or Bitcoin block, and the resulting 256 bits are used as key, to AES-256 encrypt as many immediately preceding blocks as needed, with the ciphertext used as the pseudo-random stream. The IDs of the Ethereum or Bitcoin blocks used can be provided as authentication of honest *RAND* execution.[13]
- E_K / D_K — symmetric encryption / decryption with secret key K, e.g. AES-256, or a suitable alternative.
- [TAG , *it1* , … , *itn*] — denotes a schematic data structure record, where TAG is an identifier that indicates the type (and thus purpose) of the data structure and thereby defines the format of the remaining items, and *it1* , … , *itn* are the remaining items.

The principal data structures we need for normal operation are the following. (We introduce further data structures in our discussion of punishment mechanisms below. These are left out here, for clarity.)

- [RelDdata , *RealIdentity* , *Credentials* , *SysID* , *now*] — consists of: the tag “RelDdata”, indicating that the data concerns a participant’s real identity; *RealIdentity*, the actual real identity; *Credentials*, the participant’s credentials; *SysID*, the participant’s system-generated system identity; *now*, timestamp of the creation time of the data record.
- [PUIDdata , *PUID* , *goodSt* , *now* , *exp*] — consists of: the tag “PUIDdata”, indicating that the data concerns a valid public pseudo-identity for a participant; *PUID*, a valid pseudo-identity for the participant; *goodSt*, the good standing flag of the participant; *now*, creation timestamp; *exp*, expiry time of the data record.
- [PUIDSYSdata , *PUID* , *NcrData* , *KeyPart* , *Link* , *now* , *exp*] — consists of: the tag “PUIDSYSdata”, indicating that the data concerns the encrypted *SysID* of a participant using public pseudo-identity *PUID*; *PUID*, the PUID in question; *NcrData*, an encryption of a data record [NCRSYSID , *SysID*], consisting of the tag “NCRSYSID” and *SysID*, this record being symmetrically encrypted with a key *K*; *KeyPart*, a prefix of the key *K* that decrypts the preceding encrypted record; *Link*, a pointer to the next record in a circular list of PUIDSYSdata records of *PUID*’s belonging to the same *SysID*; *now*, creation timestamp; *exp*, expiry time of the data record.
- [SNARKdata , *zkDATA* , *PlainData* , *NcrData* , *Claim*] — consists of: the tag “SNARKdata”, signifying that it contains zk-SNARK data; *zkDATA*, the succinct zero-knowledge proof digest; *PlainData*, plaintext data whose honesty *zkDATA* attests to; *NcrData*, the encrypted data whose honesty *zkDATA* attests to; *Claim*, (a reference to) the property relating *PlainData* and *NcrData* of which *zkDATA* is a zero-knowledge proof digest.
- [SMCONdata , *SmConType* , *SmConID* , *PUID1* , *PUID2* , … , *PUIDn* , *data1* , … , *datam* , *now* , *exp*] — consists of: the tag “SMCONdata”, signifying that it pertains to a smart contract instance executing on the chain; *SmConType*, the smart contract type; *SmConID*, the identifier of the specific instance of the smart contract; *PUID1* , *PUID2* , … , *PUIDn*, the public pseudo-identities of the participants engaging in this smart contract instance; *data1* , … , *datam*, other data published on-chain for this smart contract instance; *now*, the current timestamp; *exp*, the expiry timestamp.

[12] We can envisage a number of factors that can contribute to ensuring that the re-encryption is not too burdensome. One possibility would be to simply not re-encrypt at *every* committee change, but to do so only every several changes, with the frequency determined at setup time. Another would be based on the observation that provided CDS is managed properly, with regular pruning of defunct data, its overall size would not be excessive, thus rendering the re-encryption manageable. A third would be to use relatively lightweight encryption — relying on the likelihood that notice of access being posted on-chain would discourage inappropriate ‘hacking’. Lightweight possibilities include stream encrypting the data using a PRNG stream with seed chosen by the relevant chair. Then, decryption and re-encryption could be done by XORing the encrypted data with the XOR of previous and new PRNG streams. (And if the PRNG streams were CTR generated, random access would be available too — XTS-AES would do a more heavy duty job on this.) Alternatively, rather than decrypting and re-encrypting the data itself, the access rights to the encrypted data may be changed. This, though, would require storing the encrypted data itself with a trusted third party.

[13] It is envisaged that the *RAND* mechanism is different from the PRNG stream referred to in footnote 12. The purpose of the PRNG stream referred to there is to provide a means to rapid (en/de)cryption. The purpose of *RAND* is to provide an unimpeachable source of unbiased random bits.

Ovbiously, in all the above cases, where applicable, *exp* – *now* $\leq$ *MAXLIFE*.

The example data organisation just described, and the deadline structure in particular, has been designed bearing in mind that once data has been put on-chain, it remains there forever, in principle, unless specific design decisions are taken to keep the chain content manageable. This is a key factor in determining the balance between data kept on-chain and data kept off-chain.

9.3 | The Management Layer

In this section we describe the most significant management layer tasks, reserving the greatest detail for the most novel tasks, and sketching the more predictable ones more superficially.

9.3.1 | Enrollment of Participants

This section focuses on the enrollment of participants in the system and the preparations that make them ready to participate in smart contracts.

Before attempting to register with the chain, a supplier *S* chooses a (temporary, random) *L*-bit[14] private pseudo-identity *PRID_S* and computes its public pseudo-identity *PUID_S* = H^N(*PRID_S*), where the preimage index *N* is large enough. This done, supplier *S* registers with the chain using its real identity and credentials, whereupon *S* is verified in an appropriate way. The registration process as a whole is captured in the following high level procedure, discussed in more detail below.

```
TryRegisterSupplier ( S ,                                  // S's real identity
                      S's credentials ,                    // other data to verify S
                      PUID_S ,                             // S's first public pseudo-identity
                      N )                                  // preimage index of PUID_S
  require S's credentials check out
  then
  PostCDS ( [ RelDdata , S , S's credentials ,             // tag , real identity , credentials
              SysID_S ,                                    // system generated ID of S
              now ] ) ;                                    // registration timestamp
  SendTo ( S ,                                             // destination
           SysID_S ) ;                                     // system ID of S
  ReceiveFrom ( S ,                                        // source
                E_K(                                       // encryption with key K chosen by S
                    [ NCRSYSID ,                           // tag
                      SysID_S ] ) ,                        // system ID of S
                [ SNARKdata ,                              // tag
                  zkDATA ,                                 // zero-knowledge proof digest
                  KeyPart ,                                // appropriate length prefix of K
                  NcrData ,                                // encrypted SysID_S record, E_K( [ NCRSYSID , SysID_S ] )
                  Claim ] ) ;                              // zkDATA confirms that ( ∃ KeyRest • KeyPart::KeyRest decrypts NcrData )
  PostCDS ( [ PUIDSYSdata ,                                // tag
              PUID_S ,                                     // S's public pseudo-identity
              NcrData ,                                    // encrypted SysID_S record received from S, E_K( [ NCRSYSID , SysID_S ] )
              KeyPart ,                                    // appropriate length prefix of K
              Link ,                                       // pointer to next PUIDSYSdata record for PUID_S, initalised to self
              now ,                                        // creation timestamp
              exp ] ) ;                                    // expiry timestamp
  PostCHN ( [ PUIDdata ,                                   // tag
              PUID_S ,                                     // S's public pseudo-identity
              N ,                                          // preimage index of the pseudo-identity
              goodSt ,                                     // good standing flag of PUID_S, initialised to true
              now ,                                        // creation timestamp
              exp ] ) ;                                    // expiry timestamp
end
```

The pseudocode above exhibits a pattern used below. **Try...** indicates a call for some functionality that can only be allowed to change the state of the system if the contained **require** clause evaluates to **true**. If it does not, then no change of state takes place and an error code can be returned

[14] *L* = 256 would be appropriate.

to the caller if need be. If it does, the **then** clause is executed. The **then** clause is presumed to consist of primitive actions and calls that themselves do not require any non-trivial preconditions.

In the present case, assuming the credentials of *S* check out, the RelDdata record, containing the real identity information and a system ID for *S* (generated by the agent performing the registration, presumed, in our simple model to be the chair of the committee) *SysID_S*, as well as the registration timestamp, is posted to CDS.

The registration agent sends the *SysID_S* to *S*, who generates a key *K* and uses it to encrypt a data record consisting of the tag "NCRSYSID" and *SysID_S* itself. This is returned to the registration agent. To prove that the ciphertext is what it claims to be, the ciphertext is accompanied by a zk-SNARK confirming that the *KeyPart* that it contains, is indeed a prefix of the key *K* (which remains private to *S*). In other words, there is a key suffix *KeyRest*, such that the concatenation *KeyPart*::*KeyRest* indeed correctly decrypts the ciphertext.

Once the registration agent confirms the veracity of the encrypted **[** NCRSYSID , *SysID_S* **]** record the registration agent posts the PUIDSYSdata record to CDS. This associates the public pseudo-identity *PUID_S* to the real system identity *SysID_S*, obscured by *S*'s encryption. The appropriate length prefix of *K* is also recorded, so that with enough effort, the whole key can be guessed and thus *SysID_S* revealed, should it become necessary.[15] These elements constitute a simple **Obs** mechanism for our example.

As the management committee is changed regularly, and *S* subsequently generates further public pseudo-identities without revealing *SysID_S*, the privacy of *S*'s activity is recovered provided the management committees are not overwhelmingly malicious.

The PUIDSYSdata record also contains a *Link* field so that all public pseudo-identities belonging to *SysID_S* can be chained in a circular list. Should the need arise to punish their owner, with effort, they can all be retrieved, preventing their owner from escaping punishment via the use of other pseudo-identities. Finally, the record contains creation and expiry timestamps.

With the **Obs** and punishment mechanisms secured in the PUIDSYSdata record, the registration agent posts a PUIDdata on-chain, announcing that *PUID_S* is (a public pseudo-identity belonging to) a legitimate supplier, along with the relevant preimage index, the good standing flag associated to *PUID_S*, initialised to **true**, and timestamps which are identical to those in the PUIDSYSdata record.

9.3.2 | More Pseudo-Identities

As just indicated, once a participant *S* is registered and has one public pseudo-identity, more pseudo-identities can be generated for *S*, helping to conceal *S*'s activities from scrutiny by competitors. The mechanism for this is as follows.

Let *PUID_X* be an existing PUID for *S*. This implies that *S* is in possession of the relevant system ID *SysID_S* for *S*, and of the corresponding private pseudo-identity *PRID_X*, preimage index *N_X*, and key *K_X* used to encrypt **[** NCRSYSID , *SysID_S* **]** in the creation of the PUIDSYSdata record for *PUID_X*. Suppose *S* now wants to create a new PUID. To do this, *S* creates a new private pseudo-identity *PRID_new* and preimage index *N_new*, and calculates the new public pseudo-identity $PUID_new = H^{N_new}(PRID_new)$. The rest of the process is then similar to what was described before.

TryNewPUID (*PUID_X* , // existing PUID
preID_X , // candidate private ID of *PUID_X*
M_X , // $preID_X = H^{(N_X - M_X)}(PRID_X)$
PUID_new , // proposed new PUID
N_new) // preimage index of *PUID_new*

require *goodSt* flag of *PUID_X* is **true** **and**
$M_X < N_X$ **and** $PUID_X = H^{M_X}(preID_X)$ **and**
no earlier on-chain transaction has used a candidate private ID, *M'_X*, for *PUID_X* such that $M_X \leq M'_X$ **and**
current time is no later than the expiry time of *PUID_X*
then
ReceiveFrom (*PUID_X* , // source
E_{K_new}(// encryption with key *K_new* chosen by *S*
[NCRSYSID , // tag
SysID_S **]**) , // system ID of *S*
[SNARKdata , // tag
zkDATA_new , // new zero-knowledge proof digest
KeyPart_new , // appropriate length prefix of *K_new*
NcrData_new , // newly encrypted *SysID_S* record from *S*, E_{K_new}(**[** NCRSYSID , *SysID_S* **]**)
Claim_new **]**) ; // *zkDATA_new* confirms that
// (∃ *KeyRest_new* , *KeyRest_X* •
// $D_{(KeyPart_new::KeyRest_new)}$(*NcrData_new*) =
// $D_{(KeyPart_X::KeyRest_X)}$(*NcrData_X*))
// [i.e. both old and new encryptions are of the same *SysID_S* record]

[15] The presence of the fixed tag NCRSYSID in the encrypted record permits the correct key to be identified among all the guesses of a brute force search.

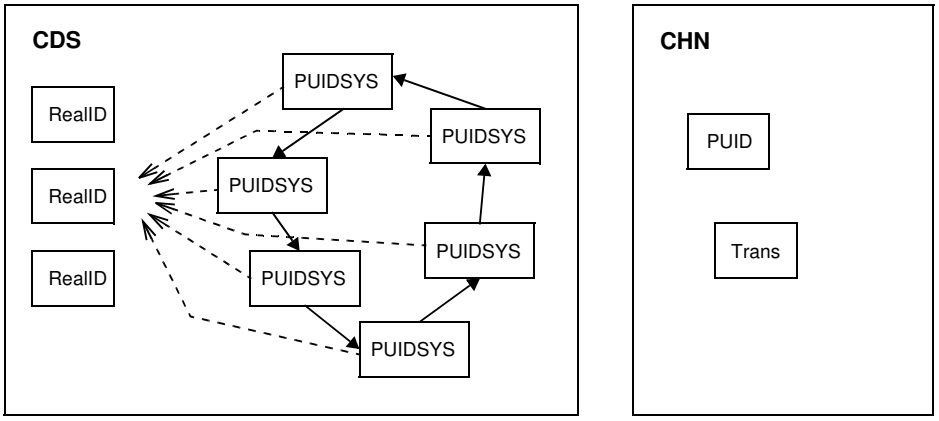

FIGURE 3 The conventional data store and blockchain in the *PayLck* smart contract system.

PostCDS (**[** PUIDSYSdata , // tag
PUID_new , // *S*'s new public pseudo-identity
E_{K_new}(**[** NCRSYSID , *SysID_S* **]**) , // encrypted *SysID_S* record received from *S*
KeyPart_new , // appropriate length prefix of *K_new*
Link_new , // copy of *Link_X* pointer from PUIDSYSdata record for *PUID_X*
now , // creation timestamp
exp **]**) ; // expiry timestamp

UpdateCDS (**[** PUIDSYSdata , // tag
PUID_X , // *PUID_X* as before
E_{K_X}(**[** NCRSYSID , *SysID_S* **]**) , // as before
KeyPart_X , // as before
Link_update , // pointer to PUIDSYSdata record for *PUID_new*
now_X , // as before
exp_X **]**) ; // as before

PostCHN (**[** PUIDdata , // tag
PUID_new , // *S*'s new public pseudo-identity
N_new , // preimage index of the pseudo-identity
goodSt_new , // good standing flag of *PUID_new*, initialised to **true**
now , // creation timestamp
exp **]**) ; // expiry timestamp

end

In the above, each putative participant *X* to the contract asserts its ownership of the public pseudo-identity *PUID_X* it is using for this purpose by offering a preimage *preID_X* (along with the relevant preimage index *M_X*) of its public pseudo-identity *PUID_X* as a signature. Of course, the preimage index *M_X* must be greater than any preimage index previously used to sign this public pseudo-identity, and must be less than the preimage index given at registration, both facts being checked in the **require** clause. This done, provided the good standing flag of *PUID_X* is **true** and the lifetime of *PUID_X* has not expired, the new public pseudo-identity can be accepted. The committee chair receives from *PUID_X* the **[** NCRSYSID , *SysID_S* **]** record, encrypted this time with a new key *K_new*, and a prefix *KeyPart_new* of *K_new*. Along with this is sent a zk-SNARK confirming that a completion *KeyRest_new* of *K_new* exists, such that the concatenation *KeyPart_new*::*KeyRest_new* decrypts the newly encrypted **[** NCRSYSID , *SysID_S* **]** record to the same plaintext to which the concatenation *KeyPart_X*::*KeyRest_X* decrypts the encrypted record associated with *PUID_X* (where the existence of *KeyRest_X* was confirmed previously, although its value remains obscure to all but *S*).

Once the committee chair has verified the zk-SNARK, the PUIDSYSdata records are suitably amended. Firstly, a new record is introduced for *PUID_new*. It contains the expected data items, with its *Link_new* being a copy of the link in the existing PUIDSYSdata record for *PUID_X*. Secondly, the *Link_X* in the PUIDSYSdata record for *PUID_X* is updated to point to the new PUIDSYSdata record for *PUID_new*, thus inserting the latter into the circular list. Finally, the details of *PUID_new* can be posted on-chain.

It is relatively clear from this description, that unless there is dishonesty in the creation of the initial public pseudo-identity, and there is collusion among committee members and between committees, the creation of new public pseudo-identities can be accomplished without finding out the true identity of the owner of the new pseudo-identity. Fig. 3 illustrates the loose connection between real identity information and pseudo-identity information in the conventional data store, and the lack of any such connection in the chain itself.

We have focused above on the *S*uppliers. 'Heavyweight' customers can be registered in the same manner. 'One off' customers might employ a more lightweight mechanism in which the relevant supplier registers a public pseudo-identity (that the customer has created) on the customer's behalf. Moreover, we have not imposed any limit on the number of pseudo-identities that a single participant might create. The proliferation of pseudo-identities obviously helps to obscure the activities of any single participant, but, as we noted already, also leaves the door open for denial of service insider attacks. However, the restricted scope of the permissioned chain is intended to bias behaviour to disincentivise participants from launching such an attack, and variations of the protocol (that keep track of how many pseudo-identities are associated with a single real identity[16]) may be designed to mitigate the possibility. For simplicity, we do not attempt to address such variants of the protocol in the present description, though they could obviously be contemplated in a more realistic design.

9.3.3 | A Simple Punishment Mechanism in Outline

One of the keystones of the PnR architectural design is the possibility of imposing punishments on miscreants in the application ecosystem. The application model we have introduced is very simple, and sufficiently unconnected with any specific application domain (which might point to specific issues to monitor), that it is not immediately obvious what constitutes bad behaviour in our model (aside from the self-evident case of generic failure to perform management committee or chair roles). Nevertheless, we can make some widely applicable observations.

Given that we envisage the system under discussion to be permissioned, an entity may conceivably encounter issues with enrollment.[17] The permissioned nature of the system implies that there is a public facing non-chain facet of the system for new entrants to engage with in the first instance. Problems arising there can be handled as they would be in the case of any similar issue between an entity and the public facing part of an organisation, thus are outside the scope of this discussion.

Focusing on the on-chain part of the system, we start with the observation, well known from the fault tolerant world, that it is impossible to guard against *every conceivable* bad outcome, no matter how extreme, since there is always some conspiracy that one could imagine that could outwit any finite set of defences. So we will restrict ourselves to discussing a three simple scenarios. First though, a couple of additional remarks.

We normally expect that successive actions occur within the timespan allowed by any predecessor action that they depend on, or that enabled them. This is tacitly implied by the informal overview of *PayLck* earlier, as well as by the handling of the enrollment and additional pseudo-identity protocols above. It will be reinforced by the more detailed description of smart contract execution in Section 9.4. It follows that if there are deadlines, there must be provisions for dealing with deadline expiry.

In *PayLck*, as well as 'normal' transitions that progress the state machine of Fig. 2 within the lifetime of the current state, there are 'expiry' transitions that react to the expiry of the current state's permitted duration. In a state machine model like *PayLck*, it is relatively easy to statically determine whether every state has a transition to deal with the expiry of its permitted duration. However, even when this is the case, there is no guarantee at runtime that the participant responsible for executing this transition will do so. Therefore, the system itself must back up in-machine deadline expiry handling with an out-of-machine mechanism of some kind. This leads us into potential punishment territory.

We will see in Section 9.4 that attempts by participants to run the *PayLck* state machine improperly have been defined to either have a null effect (no change of state) or to result in *ExceptionExit* transitions that cause exit from the on-chain protocol and resolution by human mediated procedures. To keep things simple, we will not regard such things as necessitating punishment in the present account (although, obviously, a different view could be taken). This leaves the out-of-machine handling of deadline expiry cases as the source of our punishment examples.

We will thus assume that along with the contract-determined or system-determined deadline for any action, there is a longer, *backstop deadline*, upon the expiry of which, the system will take unilateral action to punish the non-timely prosecution of the relevant protocol.

The existence of backstop deadlines implies the existence of backstop timers to monitor them. These are run by the members of the committee (or a suitable subset of them) — let us call them the backstop timer subcommittee (BTSC). When a new block is created by the committee chair (and is verified by committee members), each member of BTSC starts a backstop timer for each of the new states confirmed by the transactions in the block, and cancels the backstop timers for all the states that *enabled* the transactions in the block, which are still running up to that point.

Should a BTSC member observe that a backstop timer has expired before being canceled, it attempts to post a **[** BSTIMEXP , *details* **]** (backstop timer expired) data structure package to the chain. We assume that faults and misbehaviour are sufficiently rare in the PnR blockchain that an adequate majority of the committee receive this package into their copy of the pending transactions queue. Since blockchain participants will, generally speaking, be approximately if not precisely synchronised in time, we can expect a number of instances of this package to be generated and received. Upon receipt of such a package, any member who would be due to send one itself but has not yet done so desists from sending it, and duplicates which are received are eliminated. The BTSC members in receipt of the package elect a chair to handle the situation (which need not be the existing committee chair). What happens next depends on the nature of the transgression that has taken place.

[16] With the data structures we have outlined, this could be done without the necessity of revealing the real identity in question.

[17] One example of this might be a real participant attempting to enroll multiple times, in order to have multiple disjoint circular lists of PUIDSYS records in CDS, enabling misbehaviour using one group of PUIDs while retaining other, completely separate groups of PUIDs untainted.

Example 1. A smart contract has not progressed in a timely manner. The smart contract's interrupted operation affects only the participants involved (and there will usually only be two of them). Their participation in the contract was undertaken voluntarily, and in English law, is governed by the *caveat emptor* principle. Accordingly, the only fault, as far as the PnR system is concerned, is the possibility of leaving data pertaining to the contract lying around in the blockchain longer than the relevant deadlines would allow. Thus the punishment meted out is relatively slight. The (PUID of the) miscreant participant is identified, and the *GoodSt* flag in its PUIDSYS record is unset. This prevents further activities linked to that PUID from taking place, and existing activities linked to that PUID will be able to expire according to normal deadline provisions. The punishment in this case is slight since, as described above, the PUID's real owner will typically have other PUIDs available under which it can continue to do business. When the relevant PUID's lifetime has expired, it can be deleted in the normal way.

Example 2. A non-chair committee member has neglected to perform block verification (and/or similar) duties in a timely manner. Guilty committee members, who undermine the smooth running of the PnR blockchain, deserve stronger punishment than ordinary participants. In this case the PUID of the miscreant committee member is identified, and, as well as unsetting the *GoodSt* flag of that PUID in the relevant PUIDSYS record, the pointers in the PUIDSYS record that lead to the *whole* of the circular list of PUIDSYS records of PUIDs belonging to that PUID's owner are identified, and their *GoodSt* flags are unset too. This imposes a more severe curtailment of that participant's activities than in the previous case. The miscreant is prevented from engaging with the chain, although this is managed in a fairly discreet manner. (The said miscreant may be permitted to discreetly reapply for full participation rights after the elapse of a sufficient sentence, if suitable mechanisms were to be put in place.)

Example 3. The committee chair has neglected to perform block creation (and/or similar) duties in a timely manner. We reserve the most punitive measures available in this simple PnR model for malevolent committee chairs. This time the punishment extends to public shaming. So, the PUID of the miscreant committee chair is identified, and, as well as all PUIDs connected with that PUID's real identity being disabled (as in the previous example), the real identity is revealed. To do this requires the brute force breaking of the encrypted link back from the relevant PUIDSYS records to their owner's RealID record.

At the time he was enrolled in the PnR application system, the miscreant committee chair used some key to encrypt his SysID, and sent a prefix of the key to the then chair, whereupon it was verified using the accompanying zk-SNARK. Now, the space of bitstrings which are potential suffixes of the given prefix is partitioned among the members of the BTSC, who work on the brute force breaking until the right suffix is found. As discussed in Section 9.3.4, this will typically require significantly more computing resources than are needed for day to day operation of the PnR system, and provisions for such exceptional needs must be built into participants' obligations at enrollment time. Assuming all this is in place, the brute force breaking of the encryption can be done.[18] Once the encryption has been broken, the real identity in question is revealed, and a data package announcing that participant's real real identity and disbarment from the PnR ecosystem can be posted to the chain.

Evidently, the above constitutes a very crude proposal for the possibilities for PnR blockchain application governance, based on the very limited information about participants' behaviour that is available to us. This, in turn, is a consequence of the very limited data regarding participants' behaviour that we have included in our model (namely, the *GoodSt* flags of individual participants, and the knowledge about what is expected of them which follows from their role (i.e., ordinary participant, non-chair committee member, or committee chair). Even so, we have seen that it is possible to use even this limited data to impose a range of different sanctions, as befits the situation at hand. It is clear how the mechanisms just described constitute denial of service and ultimately revocation of anonymity for miscreants. Obviously, more nuanced governance models would be possible with the inclusion of richer data pertinent to participants' behaviour in the PUIDSYS records held in CDS.

9.3.4 | Other Management Layer Issues

The management layer tasks earlier were treated in fair detail because they typify the most innovative elements of the *PayLck* application (and the elements where further innovation would be most anticipated). In this section we discuss other management layer responsibilities, but we do so more briefly, since these are mostly relatively routine.

Parameters. Many elements of the account above were lacking, in that their more precise properties depended on parameters, for which the most appropriate values were not investigated. Among these we could list: the value of *MAXLIFE*; the size of PUID preimage values *N_Z*; size of key prefix *KeyPart_Z*; duration of PUIDs; duration of smart contracts; resource implications of the use of zero-knowledge techniques; committee size; committee/chair re-assignment frequency; frequency of archiving of on-chain data; and so on. We do not explore these issues in this paper, though they would need to be evaluated seriously for an actual implementation.

Initialisation. We assume we start with an initialised but empty system created by the application's creating agent or initiating consortium. Ideally, there would be an initiating consortium large enough for the anonymisation techniques inherent in the use of public pseudo-identities to be immediately effective. But we cannot simply guarantee such a thing, or advise against the creation of PnR applications where it is not

[18] In effect, this is a kind of Proof of Work approach to the most severe kind of punishment that is contemplated in the present incarnation of the PnR architectural model.

the case from the get go. In such cases, part of the parameter evaluation that has to take place is the impact of small initial consortia on the anonymisation properties of the application. And this impact will itself be application dependent. We mentioned above the possibility of guarding against retrospective manipulation of the chain in the small consortium initial stages of a PnR application by regularly recording the hash of the most recent PnR block on a large well established blockchain like Ethereum.[19] This would appear to be a sensible precaution to employ.

Committee/Chair Reassignment. Assuming an appropriate frequency for committee/chair re-assignment has been identified, a transparent mechanism for doing the reassignment would be needed. The *RAND* mechanism would be employed to generate a stream of pseudo-random bits, and these would be utilised by a publicly known deterministic process to generate PUIDs of the new committee and chair. To impede the easy subversive tracking of PUID owners by committee members, it would be desirable for successive committees to be disjoint. Once the new committee/chair had been decided, the access to off-chain data that overall application management requires would be reassigned by the creation of new keys. If necessary, the veracity of the process for the creation of these could be secured by zero-knowledge techniques similar to those used above.

Deadline Management. Above, we already commented on deadline handling within a smart contract, as it is intimately connected with the punishment mechanism we described. Beyond this, we stipulate the following. To permit participants to stay active in the application for extended periods, a new public pseudo-identity for a participant is permitted to have (and will regularly need to have) a lifetime that exceeds the lifetime of the pseudo-identity used to create it. On the other hand, for other actions initiated by a given pseudo-identity, e.g. the initiation of a smart contract (as we have described it), the overall lifetime of the contract must not exceed that of its participating pseudo-identities. Evidently this is potentially rather restrictive, and more involved protocols may be envisaged that allow responsibilities for participants' roles in a long lived smart contract to be handed over to fresh pseudo-identities. Equally, long lived smart contracts may be broken up into shorter segments, etc. We do not explore more intricate possibilities such as these here.

Archiving. Assuming an appropriate frequency for the archiving of on-chain data has been identified, the principles governing such archiving are relatively easy to state. The value of *MAXLIFE* must not exceed (the inverse of) the archiving frequency. Also, all top level transactions must have durations not exceeding *MAXLIFE*, and the aggregated durations of all (possible sequences of) subordinate transactions of a transaction must total less than the duration of their parent transaction, and hierarchically. If this is the case, at any archival point, it is sufficient to go back in time on the chain to a point at which the *now* timestamps of transactions are older than *MAXLIFE* ago, and archive all transactions that predate that point. This, and other management layer issues, may be aided by having certain nodes maintain the entire blockchain state rather than just the sequence of blocks [111,112].

Resourcing. These days, any organisation of any size hosts a web site, and many such sites provide online versions of the organisation's services. All of this incurs costs that are viewed, these days, as routine costs of doing business in the relevant sector. The PnR scheme has been designed so that the costs, in terms of computing resources, can similarly be viewed as simply routine costs of doing business, not massively out of line with non-blockchain service provision, *provided wasteful techniques such as POW consensus are avoided*. One aspect that may somewhat stretch this quite optimistic picture, is the cost of setting up the various zero-knowledge objects needed in the various protocol elements. This may be relatively more costly than for most of the other algorithmic procedures needed for PnR. However, research into zero-knowledge techniques (particularly prompted by the demand from blockchain applications) is reducing such costs significantly, so there is little justification for gloom over this aspect.

By contrast, in the punishment discussion of Section 9.3.3 there arose the issue of utilising potentially quite significant computing resources to break the anonymity of miscreants in the application ecosystem. It would be expected that these may well go beyond what participants routinely have, or can routinely make available. Thus, the need for hiring of third party computing resources is a natural consequence of the PnR architectural model. It is envisaged that when a PnR system would be set up, contractual arrangements would be created to make such resources available when needed, e.g. via a cloud provider. Additionally, the acceptance of a new participant into the PnR system would entail contractual obligations by the new participant to support their share of the associated costs, whether on demand, or via some up front deposit. All of this would be handled quite conventionally, motivated by participants' desire to benefit from the application system.

9.4 | The Application Layer

The application layer is primarily concerned with initiating and progressing smart contracts by participants. We assume that there are a limited number of smart contract schemas that the system can execute; we assume that *PayLckSCH* is the label that distinguishes the *PayLck* schema. Any individual instance of any smart contract schema, e.g. *PayLck*, will have its own unique distinguishing ID *SmCtID*, and (for simplicity we will assume) will involve a predetermined number of participants. The schema label, instance ID, and participants will uniquely identify any instance of a given smart contract, although if instance IDs have global scope — which we will assume henceforth, then the instance ID will be sufficient for unique identification. For the rest of this section we restrict attention to *PayLck* for purposes of illustration.

[19] One PnR transaction would record the block ID and its hash that was sent to Ethereum. A subsequent PnR transaction would record the Ethereum transaction in which these details were placed.

In the case of *PayLck*, there is a supplier with public pseudo-identity *PUID_S* and a customer with pseudo-identity *PUID_C*. We will not discuss here the details that are transacted between the participants of a *PayLck* instance off-chain. We focus on the on-chain activities, which are relatively simple to describe. Assuming the required off-chain negotiations between the participants have been sufficiently well completed that the participants wish to commit to the smart contract, initiating a *PayLck* contract works as follows.

TryInitiateSmartContract (*PayLckSCH* , // contract type
PUID_S , // PUID of contract originator *S*
preID_S , // candidate private ID of *PUID_S*
M_S , // $preID_S = H^{(N_S - M_S)}(PRID_S)$
PUID_C , // PUID of *C*
preID_C , // candidate private ID of *PUID_C*
M_C , // $preID_C = H^{(N_C - M_C)}(PRID_C)$
LocDat , // data to be stored on-chain
IPFSref , // reference to data to be stored on IPFS
StDline , // deadline for duration of starting state
ConDline) // deadline for duration of whole contract

require *goodSt* flags of *PUID_S* and of *PUID_C* are both **true** **and**
$M_S < N_S$ **and** $PUID_S = H^{M_S}(preID_S)$ **and**
no earlier on-chain transaction has used a candidate private ID, *M'_S*, for *PUID_S* such that $M_S \leq M'_S$ **and**
$M_C < N_C$ **and** $PUID_C = H^{M_C}(preID_C)$ **and**
no earlier on-chain transaction has used a candidate private ID, *M'_C*, for *PUID_C* such that $M_C \leq M'_C$ **and**
StDline < *ConDline* **and**
ConDline is no later than the expiry times of either *PUID_S* or *PUID_C*

then

PostCHN (**[** SMCONdata , // tag
PayLckSCH , // contract type
SmConID , // system generated contract instance ID
PUID_S , // PUID of contract participant *S*
M_S , // candidate preimage index of *PUID_S*
PUID_C , // PUID of contract participant *C*
M_C , // candidate preimage index of *PUID_C*
LocDat , // on-chain data
IPFSref , // reference to IPFS data
fromPayLckState , // contract from-state, *NULL* as this is the start of the contract
TransName , // contract state machine transition name, 'Start' in the case of *PayLck*
toPayLckState , // contract to-state, 'Working' in the case of *PayLck*
now , // creation timestamp
StDline , // deadline for duration of starting state
ConDline **]**) ; // deadline for duration of whole contract

end

In the above, the similar identity checks that appeared in Section 9.3.2, are done in the **require** clause for each putative participant in the contract. Also, the expiry time of the proposed starting state should not exceed the expiry time of the overall contract, and this in turn should not exceed the lifetime of either *PUID_S* or *PUID_C*. If this all checks out, a system generated contract instance ID is created, and a SMCONdata record is posted to the chain containing the contract type, instance ID, participants (along with their signature data), any relevant on-chain and off-chain data (the latter assumed held in IPFS, encrypted with a key known only to the participants), contract start state, timestamp, starting state duration deadline and overall contract duration deadline. In the absence of dishonest behaviour, the exclusive use of public pseudo-identities protects anonymity.

Once the contract is under way, successive transitions are all handled in the same way. There is typically some offline work to be done in the current contract state which we do not delve into. Once it is completed and any required details are posted to IPFS, an on-chain transition is posted as follows.

TryTransaction (*PUID_orig* , // PUID of transaction originator
preID_orig , // candidate private ID of *PUID_orig*
M_orig , // $preID_orig = H^{(N_orig - M_orig)}(PRID_orig)$
PayLckSCH , // contract type
SmConID , // contract instance ID
LocDat , // data to be stored on-chain
IPFSref , // reference to data to be stored in IPFS
fromPayLckState , // contract from-state
TransName , // contract state machine transition name
toPayLckState , // contract to-state
StDline) // deadline for duration of to-state

require *goodSt* flag of *PUID_orig* is **true** **and**
M_orig < *N_orig* **and** *PUID_orig* = H^{M_orig}(*preID_orig*) **and**
no earlier on-chain transaction has used a candidate private ID, *M'_orig*, for *PUID_orig* such that *M_orig* $\leq$ *M'_orig* **and**
PUID_orig is one of *PUID_S* or *PUID_C*

then if *fromPayLckState* is the current state of the *SmConID* instance of *PayLck* on-chain **and**
PUID_orig is the correct participant to execute transition *TransName* in *fromPayLckState* **and**
the timing conditions for transition *TransName* in *fromPayLckState* are satisfied **and**
StDline $\leq$ *ConDline*

then

PostCHN (**[** SMCONdata , // tag
PayLckSCH , // contract type
SmConID , // system generated contract instance ID
PUID_S , // PUID of contract participant *S*
M_S , // candidate preimage index of *PUID_S*
PUID_C , // PUID of contract participant *C*
M_C , // candidate preimage index of *PUID_C*
LocDat , // on-chain data
IPFSref , // reference to IPFS data
fromPayLckState , // contract from-state for this transition
TransName , // contract state machine transition name
toPayLckState , // contract to-state for this transition
now , // timestamp
StDline , // duration of to-state
ConDline **]**) ; // duration of whole contract

else

PostCHN (**[** SMCONdata , // tag
PayLckSCH , *SmConID* , // contract type , system generated contract instance ID
PUID_S , *M_S* , *PUID_C* , *M_C* , // participant data
LocDat , // on-chain data, in this case *ExceptionExit*
LocDatDetails , // reason for the *ExceptionExit*
now **]**) ; // timestamp

SendTo (*PUID_S* , // destination
ExceptionExit , // *ExceptionExit* notification
PayLckSCH , // contract type
SmConID , // system generated contract instance ID
LocDatDetails) ; // *ExceptionExit* details

SendTo (*PUID_C* , // destination
ExceptionExit , // *ExceptionExit* notification
PayLckSCH , // contract type
SmConID , // system generated contract instance ID
LocDatDetails) ; // *ExceptionExit* details

fi

end

In the above, we see that the transaction originator *PUID_orig* is first authenticated as in Section 9.3.2 in the **require** clause, and additionally, it is confirmed that *PUID_orig* is one of the participants in the *SmConID* instance of *PayLck*. This being well, two possibilities ensue. If the current state of the *SmConID* instance of *PayLck* is as claimed by *PUID_orig*,[20] and *PUID_orig* is the appropriate participant to execute the proposed *PayLck* transition,[21] and transition *TransName* is appropriate when considering the elapsed duration of the current state of the *SmConID* instance of *PayLck*,[22] then the transition proceeds as proposed, i.e. the relevant transition data are posted as an on-chain transaction. If not, then the present design dictates an exceptional exit from automated execution (*c.f.* the last point in Section 6). To this end, an *ExceptionExit* transition is posted on-chain, and notifications are sent to all contract participants.

[20] What this means is that the *fromPayLckState* stated by *PUID_orig* is the *toPayLckState* in the most recent on-chain transaction for the *SmConID* instance of *PayLck*.

[21] It is assumed that this is implicit in the name of the proposed *PayLck* transition, although this information may be recorded by other means. In the case of *PayLck* the 'business logic' makes it clear that all the transactions are to be executed by *PUID_S*, in response to the actions (or inactions) of *PUID_C*.

[22] In the case of *PayLck*, transitions are either within the current state's declared duration, or react to the expiry of the current state's declared duration.

9.5 | Summary

Above we have described the key elements of the PnR blockchain system, and how a smart contract could be launched and run on it. The level of abstraction was rather high, in that there is plenty of scope for various kinds of optimisation in an actual implementation. We also focused on the more novel elements, ones where the inherent conflicts intrinsic to the PnR architecture were unavoidable, and covered other, more routine matters more superficially. It is clear that the mechanisms for the preservation of privacy that we presented are not immune from being circumvented in the given design, but equally, this is the area in which increased cryptographic innovation has the most to offer. However, we leave such improvements for future work.

10 | CONCLUSION

In the preceding sections, we have reviewed the pros and cons of cryptocurrencies at some length. However, as well providing a medium for a wide variety of fraudulent or illegal behaviours which have caught the public attention, cryptocurrencies also open the door to leveraging scenarios in which centralised arrangements are seen to be unfair and oppressive, due to the protection of entrenched interests at the centre of power[113]. So we cannot say that cryptocurrencies are intrinsically good or intrinsically bad. We can merely say that the different mechanisms in force for regulation and stabilisation for cryptocurrencies compared with fiat (largely absent in the crypto case), enable a different spectrum of both good and bad behaviours in the two spheres. What is evident is that the character of the long term future of cryptocurrencies is still very unclear — skeptics and adherents seem equally fervent in their views for the moment, as vividly demonstrated by the contrasting views expressed e.g. on Steemit[18] on the one hand, and e.g. by Schneier[114] or Waldo[115] on the other.

Given the very real risks associated with cryptocurrencies, from volatility and illiquidity to downright fraud, we introduced the PnR architectural model for blockchain applications that can utilise the added value that blockchains can bring in some application sphere without the use of cryptocurrencies. Throughout the account we emphasised that constructing robust blockchain applications should be seen, above all, as **application specific (dis)incentive engineering**. We stressed that an incorrect appreciation of the diverse interests that contribute to a properly working blockchain system could easily lead to its failure. We underlined the point that the intrinsic benefit of the application to its participants needs to be enough to motivate good behaviour and to act as a stabilising force.

We can see this argument as being about a kind of 'arms race' between opposing interests/incentives in the following manner. Thus, whenever a counter-interest has been identified that negates some desirable property of the system, a counter-counter-interest needs to be found that outweighs it. And so on. Any such chain of competing, oppositely oriented interests, must end in an overriding interest that is aligned with the goals of the system. Otherwise, the originally conceived system destabilises. Viewed in this light, the extreme volatility of cryptocurrencies is not all that surprising, given that it is hard to identify an obvious interest that will override all others and impose stability (in the shape of much reduced volatility). The value of a cryptocurrency is, after all, predicated on belief in what the future demand for that cryptocurrency might be[36].

In an effort to make the mechanisms in PnR self-contained, we fell back on **denial of service** to, or **revocation of anonymity** in the application as a counterbalance to *Tragedy of the Commons* style temptations to neglect or subvert communal duties to maintain the blockchain protocol, or to indulge in other kinds of unacceptable behaviour. Keeping all this under control aligns well with the use of permissioned blockchains, and the restricted field of application envisaged for such a system makes its implementation easier and cheaper, e.g. forestalling the need to use bespoke hardware, which is so often seen in cryptocurrency mining systems these days. Such ideas naturally encourage cooperation between different blockchains to address issues outside the immediate design of a given application, and we can expect the trend towards inter-chain cooperation to increase in future.

We introduced an outline software architecture to support the application architecture encapsulated in the PnR concept. As we discussed the PnR architecture in greater detail, the tension between the intrinsic desire for privacy in the blockchain and the necessity for identity management (not least to enable the PnR sanctions to be applied when necessary) became increasingly apparent, and we argued that sophisticated cryptographic mechanisms could be brought to bear to address these issues.

By the arguments laid out in Section 8.2 we do not claim that the techniques proposed yielded a cryptographically watertight mechanism.[23] The conflicts of interest inherent in the PnR concept make that impossible, probably, without the use of trusted third parties, or of mechanisms with strongly TTP-like properties. Nevertheless, increasing the cryptographic innovation applied to this problem, especially when reinforced with the requirement that any privacy breaking mechanisms need the involvement of more than one participant, with the relevant cohorts randomly selected and regularly reassigned during normal operation, could, we suggest, yield acceptable solutions that approach the desired ideal in some asymptotic sense. Instead, for now, we have focused on the interests in the wider application arena, and in particular on the absence of actual

[23] By 'cryptographically watertight', we mean the usual thing, i.e. impossible to break with feasible computational resources with more than negligible probability.

value on the PnR blockchain, to reduce to a very small level the incentive to misuse the proposed system, particularly for participants who rely significantly on maintenance of reputation to sustain their activities.

We illustrated our ideas with a particularly simple payment based case study. There, the design decision was taken that actual payment and other sensitive details need not be kept on the chain. This illustrates the view, paramount in the perspective of this paper, that such application specific design issues must be considered and evaluated at the outset of the design process. Otherwise an inappropriate decision could derail the whole system.

The payment issue also shows in microcosm, a key issue in the wider takeup of blockchain based solutions, namely, that there is a perceptible tension between the traditional and understandable desire on the part of the participants for contract details to remain confidential to themselves, and the corresponding necessity for details to be made public to enable blockchain verification. This goes beyond the intrinsic tension between desired blockchain anonymity and the needs of identity management inherent in the PnR concept itself, discussed just above. It is inevitable that bridging this impasse will also call on increasingly sophisticated cryptographic techniques, and what is considered 'good enough' from this standpoint will be another element that is application dependent. The more complex the privacy concerns and interdependencies between the different parts of the application ecosystem, the more subtle will the cryptography need to be.

Key to the preceding is the use of techniques such as zero-knowledge proofs. These do not come for free, but need some preparatory setup work for their viability. Efficient zero-knowledge techniques —the present author feels— will in future, be key to sustaining complex blockchain applications, and so the focus on research on zero-knowledge techniques at the present time is well merited. All of the above reinforces the view of the present author that the future of blockchain applications will reside in the seamless incorporation of blockchain concepts into more general distributed application architectures, rather than restricting their use in some purist cryptocurrency specific and cryptocurrency restricted way.

While many of the observations made in this paper are not necessarily novel when taken in isolation, our aim was to attempt to combine them into a generic self-supporting system in which the diverse components are composed in a mutually reinforcing and stable way. In this manner, we have indicated a forward looking roadmap for a class of applications that can benefit from the inherent blockchain concepts, such as immutability, relatively unproblematically.

For the kind of blockchain applications which we have discussed in this paper, one can even foresee a future in which the immutability properties of blockchains just mentioned may become mandated by regulation in order to bring a heightened level of transparency to particular spheres of activity. As usually happens when novel technical capabilities become available, this can be a double edged sword. In liberal, typically developed, western nations, when applied to situations in which greater transparency is an evident social good, such developments would be welcomed and would be introduced relatively unproblematically. The flipside is that, in more authoritarian regimes, the same blockchain technical capabilities could become a means of more detailed monitoring of individuals and of particular activities within society, and could thus be deployed as an adjunct mechanism for state control[116].

References

1. Distributed Ledger Technology: Beyond Block Chain . https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/492972/gs-16-1-distributed-ledger-technology.pdf.

2. Hileman G, Rauchs M. Global Blockchain Benchmarking Study. Cambridge University, Judge Business School https://www.jbs.cam.ac.uk/fileadmin/user_upload/research/centres/alternative-finance/downloads/2017-09-27-ccaf-globalbchain.pdf.

3. Davidson S, De Filipi F, Potts J. Disrupting Governance: The New Institutional Economics of Distributed Ledger Technology. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2811995.

4. Wilson S. Blockchain: Almost Everything You Read Is Wrong. 2016. https://www.constellationr.com/blog-news/blockchain-almost-everything-you-read-wrong.

5. New Satesman . How Bitcoin Resembles the South Sea Bubble. https://www.newstatesman.com/politics/economy/2017/12/how-bitcoin-resembles-south-sea-bubble.

6. Bitcoin's insane energy consumption, explained . https://arstechnica.com/tech-policy/2017/12/bitcoins-insane-energy-consumption-explained/.

7. Lewis A. Avoiding Blockchain for Blockchains Sake: Three Real Use Case Criteria. 2017. https://bitsonblocks.net/2017/07/24/avoiding-blockchain-for-blockchains-sake-three-real-use-case-criteria.

8. Smith M. The Blockchain Challenge Nobody is Talking About. 2017. http://usblogs.pwc.com/emerging-technology/the-blockchain-challenge.

9. Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf.
10. Narayanan A, Bonneau J, Felten E, Miller A, Goldfeder S. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press . 2016.
11. Antonopoulos A. *Mastering Bitcoin*. O'Reilly . 2017.
12. Bitcoin Energy Consumption Index . https://digiconomist.net/bitcoin-energy-consumption/.
13. Varoufakis Y. *And the Weak Suffer What They Must?: Europe, Austerity and the Threat to Global Stability*. Vintage . 2017.
14. Stiglitz J. *The Euro: And its Threat to the Future of Europe*. Penguin . 2017.
15. Google search: cryptocurrency mining rig . .
16. Bitmain . https://www.bitmain.com/ https://en.wikipedia.org/wiki/Bitmain.
17. Bitmain . https://en.wikipedia.org/wiki/Bitmain.
18. Steemit . https://steemit.com.
19. Dale R. *The First Crash: Lessons from the South Sea Bubble*. Princeton University Press . 2016.
20. Dash M. *Tulipomania: The Story of the World's Most Coveted Flower and the Extraordinary Passions it Aroused*. W&N . 2010.
21. Great Britain. Ministry Of Transport. Financial And Statistical Dept . *Railway Returns: Returns of the Capital, Traffic, Receipts, and Working Expenditure, Etc, of the Railway Companies of Great Britain*. Ulan Press . 2012.
22. Chancellor E. *Devil Take the Hindmost: A History of Financial Speculation*. Plume Books . 1998.
23. Mackay C. *Extraordinary Popular Delusions and the Madness of Crowds*. Wordsworth Editions . 1995.
24. McMillan R. The Inside Story of Mt. Gox, Bitcoin's $460 Million Disaster. https://www.wired.com/2014/03/bitcoin-exchange/.
25. Falkon S. The Story of the DAO — Its History and Consequences. https://medium.com/swlh/the-story-of-the-dao-its-history-and-consequences-71e6a8a551ee.
26. Chainalysis Crypto Crime Report 2019 . https://blog.chainalysis.com/2019-cryptocrime-review.
27. Cryptocurrency and Criminality: The Bitcoin Ppportunity . *The Police Journal: Theory, Practice and Principles* 2016; 89(4): 327-339.
28. Meiklejohn S, Pomarole M, Jordan G, et al. A Fistful of Bitcoins: Characterizing Payments among Men with no Names. In: ACM; 2013: 127-140.
29. FXEMPIRE . How Can Bitcoin be used as a Crime Weapon? And How Can this Be Solved?. https://www.fxempire.com/education/article/can-bitcoin-used-crime-weapon-can-solved-512046.
30. Silk Road Online Drug Market Taken Down . *Network Security* 2013; 2013(10): 1-2.
31. How the Feds Took Down the Silk Road Drug Wonderland . https://www.wired.com/2013/11/silk-road.
32. The Silk Road takedown shows how the Feds can get around crypto . https://www.pri.org/stories/2016-07-14/silk-road-takedown-shows-how-feds-can-get-around-crypto.
33. Cointelegraph . Bitcoin No Longer Desirable Option for Criminals, Says CoinCorner CEO. https://cointelegraph.com/news/bitcoin-no-longer-desirable-option-for-criminals-says-coincorner-ceo.
34. Channel 4 TV . Meet The Drug Lords: Inside The Real Narcos. 2,8,15 August 2018. UK Terrestrial TV: Channel 4.
35. Cryptocurrency market capitalisation . https://coinmarketcap.com/.
36. Carney M. The Future of Money. Bank of England; . https://www.bankofengland.co.uk/-/media/boe/files/speech/2018/the-future-of-money-speech-by-mark-carney.pdf?la=en&hash=A51E1C8E90BDD3D071A8D6B4F8C1566E7AC91418.
37. Kwiat T. Beyond Bitcoin: Issues in Regulating Blockchain Transactions. *Duke Law Journal* 2015; 65: 569-608.
38. UK House of Commons Treasury Committee Report: Crypto-Assets . https://publications.parliament.uk/pa/cm201719/cmselect/cmtreasy/910/910.pdf.
39. Carney M. The Growing Challenges for Monetary Policy in the Current International Monetary and Financial System. Bank of England; . https://www.bankofengland.co.uk/-/media/boe/files/speech/2019/the-growing-challenges-for-monetary-policy-speech-by-mark-carney.pdf?la=en&hash=01A18270247C456901D4043F59D4B79F09B6BFBC.
40. Ethereum . https://www.ethereum.org/.
41. Solidity Documentation . https://solidity.readthedocs.io.
42. Solidity Github . https://github.com/ethereum/solidity.
43. Solidity . https://en.wikipedia.org/wiki/Solidity.

44. Abrial JR. *Modeling in Event-B: System and Software Engineering*. CUP . 2010.

45. RODIN Tool . In: http://www.event-b.org/ http://sourceforge.net/projects/rodin-b-sharp/.

46. Stanford Blockchain Conferences . 2017 onwards. https://cyber.stanford.edu/sbc19.

47. ERC20 replacement . https://cointelegraph.com/news/top-ethereum-token-protocols-which-may-replace-erc20.

48. Rauchs M, Glidden A, Gordon B, et al. Distributed Ledger Technology Systems: A Conceptual Framework Report. Cambridge University, Judge Business School https://www.jbs.cam.ac.uk/fileadmin/user_upload/research/centres/alternative-finance/downloads/2018-08-20-conceptualising-dlt-systems.pdf.

49. Wattenhofer R. *The Science of the Blockchain*. Inverted Forest Publishing . 2016.

50. Buterin V. On Stake. 2014. https://blog.ethereum.org/2014/07/05/stake.

51. Consensus: Immutable Agreements for the Internet of Value . 2016. https://assets.kpmg.com/content/dam/kpmg/pdf/2016/06/kpmg-blockchain-consensus-mechanism.pdf.

52. HPB . https://hpb.io.

53. EOS . https://eos.io.

54. Elastos . https://www.elastos.org.

55. DEXON . https://dexon.org.

56. Cook, P. . How Perverse Incentives are Ruining Healthcare. *rheumatology network*. http://www.rheumatologynetwork.com/healthcare-policy/how-perverse-incentives-are-ruining-healthcare.

57. Paton C. Present Dangers and Future Threats: Some Perverse Incentives in the NHS Reforms. *BMJ* 1995; 310: 1245-1248.

58. Target pregnancy story . http://www.edventure-ga.com/apsistat/Activities/Articles/How%20Target%20Figured%20Out%20A%20Teen%20Girl%20Was%20Pregnant%20Before%20Her%20Father%20Did%20-%20Forbes.pdf.

59. Hardin G. The Tragedy of the Commons. *Science* 1968; 162: 1243-1248.

60. Jones A. *Game Theory: Mathematical Models of Conflict*. Ellis Horwood . 1980.

61. Mesterton-Gibbons M. *An Introduction to Game-Theoretic Modelling*. Addison-Wesley . 1992.

62. Banach R. Punishment not Reward: Disincentivising Blockchain Application Misbehaviour. In: Proc. ICBC-19, IEEE; 2019: 185-187.

63. The EU General Data Protection Regulation . 2018. https://www. eugdpr.org/ https://gdpr-info.eu/ https://ec.europa.eu/commission/priorities/justice-and-fundamental-rights/data-protection/2018-reform-eu-data-protection-rules_en.

64. The California Consumer Privacy Act . 2018. https://www.caprivacy.org/ https://searchsecurity.techtarget.com/blog/Security-Bytes/Is-the-new-California-privacy-law-a-domestic-GDPR https://www.firstsanfranciscopartners.com/blog/california-consumer-privacy-act-of-2018-vs-gdpr/ https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375.

65. Lamport L. The Byzantine Generals Problem. *ACM TOPLAS* 1982; 4: 382-401.

66. Delegated Proof of Stake . https://coincentral.com/what-is-delegated-proof-of-stake-exploring-the-consensus-algorithm/ https://www.mycryptopedia.com/delegated-proof-stake-dpos-explained/.

67. Proof of Authority . https://medium.com/poa-network/proof-of-authority-consensus-model-with-identity-at-stake-d5bd15463256 https://wiki.parity.io/Proof-of-Authority-Chains.

68. Algorand . 2018. https://www.algorand.com/ https://medium.com/algorand/secure-blockchain-decentralization-via-committees-7602f598a0a9 https://medium.com/algorand/algorands-instant-consensus-protocol-e66ac5807e37 https://eprint.iacr.org/2018/377.pdf.

69. Hyperledger . https://www.hyperledger.org.

70. Hyperledger fabric . https://www.hyperledger.org/projects/fabric.

71. Catalyst Network . https://catalystnet.org/.

72. Jelurida . https://www.jelurida.com/.

73. Allison I. IBM and Maersk Struggle to Sign Partners to Shipping Blockchain. https://www.coindesk.com/ibm-blockchain-maersk-shipping-struggling.

74. RSK . https://www.rsk.co/en/.

75. Lightning Network . https://lightning.network/.

76. Ethereum Constantinople and St. Petersburg hard forks . https://cryptopotato.com/coming-up-constantinople-ethereum-is-getting-ready-to-todays-expected-2-hard-forks/.

77. Interplanetary File System . https://ipfs.io/ https://en.wikipedia.org/wiki/InterPlanetary_File_System.

78. Mao W. *Modern Cryptography Theory and Practice*. Prentice Hall . 2004.

79. Goldreich O. *Foundations of Cryptography, Vol I*. Cambridge University Press . 2006.

80. Zero-knowledge proof . https://en.wikipedia.org/wiki/Zero-knowledge_proof.

81. Goldreich, O. and Oren, Y. . Definitions and Properties of Zero Knowledge Proof Systems. tech. rep., Technion, C.S. Dept., Tech. Rep. 0610; 1990. http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/1990/CS/CS0610.pdf.

82. Goldreich, O. and Micali, S. and Widgerson, A. . Proofs that Yield Nothing but their Validity or All Languages in NP have Zero-Knowledge Proof Systems. tech. rep., Technion, C.S. Dept., Tech. Rep. 0615; 1990. http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/1990/CS/CS0615.pdf.

83. it q. https://qed-it.com/.

84. appliedblockchain . https://appliedblockchain.com/.

85. What are SNARKs . https://crypto.stackexchange.com/questions/19884/what-are-snarks.

86. Mayer H. zk-SNARK explained: Basic Principles. CoinFabrik; . https://blog.coinfabrik.com/wp-content/uploads/2017/03/zkSNARK-explained_basic_principles.pdf.

87. Gennaro R, Gentry C, Parno B, Raykova M. Quadratic Span Programs and Succinct NIZKs without PCPs. Cryptology ePrint Archive; 2012. https://eprint.iacr.org/2012/215.pdf.

88. BenSasson E, Chiesa A, Tromer E, Virza M. Succinct Non-Interactive Zero Knowledgefor a von Neumann Architecture. Cryptology ePrint Archive; 2019. https://eprint.iacr.org/2013/879.pdf.

89. Rackoff C, Simon D. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In: . Vol. 576. Springer, LNCS; 1992: 433-444.

90. Zcash . https://z.cash/.

91. Monero . https://monero.org.

92. ETH Zero-Proof Prototype Models: Has Ernst & Young Done What Ethereum Could Not? . https://cointelegraph.com/news/eth-zero-proof-prototype-models-has-ernst-young-done-what-ethereum-could-not.

93. Beimel A. Secret-Sharing Schemes: A Survey. In: . Vol. 6639. Springer, LNCS; 2011: 11-46.

94. Stinson D. An Explication of Secret Sharing Schemes. *Des. Codes Cryptography* 1992; 2: 357-390.

95. Brickell E. Some Ideal Secret Sharing Schemes. In: . Vol. 434. Springer, LNCS; 1990: 468-475.

96. Group signature . https://en.wikipedia.org/wiki/Group_signature.

97. Ateniese G, Camenisch J, Joye M, T. G. A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. In: . Vol. 1880. Springer, LNCS; 2000: 255-270.

98. Boneh D, Boyen X, Shacham H. Short Group Signatures. In: . Vol. 3152. Springer, LNCS; 2004: 41-55.

99. Ring signature . https://en.wikipedia.org/wiki/Ring_signature.

100. Rivest R, Shamir A, Tauman Y. How to Leak a Secret. In: . Vol. 2248. Springer, LNCS; 2001: 552-565.

101. Bresson E, Stern J, Szydlo M. Threshold Ring Signatures and Applications to Ad-hoc Groups. In: . Vol. 2442. Springer, LNCS; 2002: 465-480.

102. Multisignature . https://en.wikipedia.org/wiki/Multisignature.

103. Boneh D, Boyen X, Shacham H. Short Group Signatures. In: . Vol. 4377. Springer, LNCS; 2007: 145-162.

104. Hwang S, Chen C. New Multi-Proxy Multi-Signature Schemes. *Applied Mathematics and Computation* 2004; 147: 57-67.

105. Bogetoft P, Christensen D, Damgard I, et al. Secure Multiparty Computation Goes Live. In: . Vol. 5628. Springer, LNCS; 2009: 325-343.

106. Cramer R, Damgard I. Multiparty Computation, an Introduction. In: Springer, Birkhauser Verlag; 2005: 41-87.

107. Homomorphic encryption . https://en.wikipedia.org/wiki/Homomorphic_encryption.

108. Freedman R. *Introduction to Financial Technology*. Academic . 2006.

109. Financial technology . https://en.wikipedia.org/wiki/Financial_technology.

110. Oraclize . http://www.oraclize.it.

111. Vitalik weighs in on Ethereum post-hardfork troubles with “big scary nodes” . https://www.chepicap.com/en/news/8083/vitalik-weighs-in-on-ethereum-post-hardfork-troubles-with-big-scary-nodes-.html.

112. Ethereum insiders deny there’s only one computer keeping a copy of its blockchain . https://thenextweb.com/hardfork/2019/03/13/ethereum-insiders-fire-back-nodes/.

113. Against the 'Putrid' Euro . https://cointelegraph.com/news/against-the-putrid-euro-naples-mayor-plans-to-launch-autonomous-cryptocurrency.

114. Schneier B. Blockchain and Trust. https://www.schneier.com/blog/archives/2019/02/blockchain_and_.html.

115. Waldo J. A Hitchhiker's Guide to the Blockchain Universe. https://queue.acm.org/detail.cfm?id=3305265.

116. Bogost I. Cryptocurrency Might be a Path to Authoritarianism. https://www.theatlantic.com/technology/archive/2017/05/blockchain-of-command/528543/.

How to cite this article: R. Banach (9999), Blockchain Applications Beyond the Cryptocurrency Casino: The Punishment not Reward (PnR) Blockchain Architecture, *Concurrency and Computation: Practice and Experience*, *9999*;*12*:*345-678*.